VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

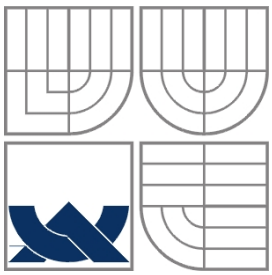
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SYNTÉZA TEXTUR

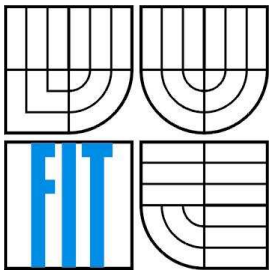
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR ONDŘEJ RABIŠKA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SYNTÉZA TEXTUR
TEXTURE SYNTHESIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ RABIŠKA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ONDŘEJ ŠILER

BRNO 2009

Abstrakt

Cílem této bakalářské práce bylo vytvořit aplikaci, která bude syntetizovat textury. Tato metoda generování syntetických textur je převážně postavena na metodě Li-Yi Weye a Marca Levoye [3]. Metoda této bakalářské práce je schopna bez větších ztrát kvality syntetizovat široké spektrum textur. Je to vysoce výpočetně náročný proces, jehož velkou nevýhodou je doba výpočtu. Proto jsem jako prostředí zvolil jazyk C++. V této bakalářské práci jsem navrhl a vytvořil dvě metody urychlení, které se dají navzájem kombinovat.

Abstract

The goal of this bachelor's thesis was to developed an application to synthesize textures. A method for the texture generation is mainly based on the work of Li-Yi Wey and Marc Levoy's [3]. This method is able to quickly and without bigger wastes of quality synthesize broad-spectrum of textures. It is a computational-intensive whose big disadvantage is the computing time. Therefore I chose the C++ programming. In this bachelor's thesis I suggested and created two acceleration method, which are possible to combine together.

Klíčová slova.

Počítačová grafika, textura, syntéza textur.

Keywords

Computer graphics, texture, texture synthesis.

Citace

Rabiška Ondřej: Syntéza textur, bakalářská práce, Brno, FIT VUT v Brně, 2009

Syntéza textur

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Šilera
Další informace mi poskytli Ing. Miroslav Švub.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Rabiška
14.5.2009

Poděkování

Velice rád bych poděkoval Ing. Švubovi za uvedení do problematiky syntézy textur a zejména vedoucímu bakalářské práce Ing. Ondřeji Šilerovi za vysvětlení principu algoritmizace syntézy textur a poskytnutí kvalitních studijních materiálů. Oběma bych chtěl poděkovat za perfektní spolupráci.

Obsah

1 Úvod

Počítačová grafika je obor zabývající se analýzou nebo tvorbou čili syntézou a generováním grafické obrazové informace. Pokud je k dispozici obrazová informace získaná měřením (kamerou, fotoaparátem), je třeba ji dále zpracovávat (upravit vylepšit) nebo analyzovat (interpretovat a rozpoznat objekty). Jedná se o aplikace typu *Počítačové vidění* [1]. Máme-li naopak k dispozici virtuální počítačový (matematický) popis (model) objektů, potřebujeme ho převést na grafickou obrazovou informaci (syntéza) a zobrazit[11].

Počítačová grafika je velmi významná, protože vyjádření grafické informace (její zobrazení) je pro člověka ve srovnání s např. textovým popisem nebo zvukovým vjemem nejvíce informačně obsažné. Lidský mozek je schopen grafickou informaci velmi rychle a detailně analyzovat a interpretovat. U jiných forem vyjádření informace má člověk sklony k dezinterpretacím nebo klamům, které se samozřejmě nevyhnou lidskému vidění [11].
Počítačovou grafiku lze rozdělit z různých hledisek:

- Podle dimenze zpracovávané informace například na 2D a 3D
- Podle oblasti použití například na zpracování fotek, videa, malování nebo CAD, vizualizaci animaci atd.
- Z hlediska uživatele grafických programů nebo z hlediska jejich programátora.

Tento článek se týká počítačové grafiky z pohledu programátora, z hlediska popisu (reprezentace) zobrazovaných dat. Z tohoto úhlu pohledu se grafika dělí na rastrovou a vektorovou[11]. Tato kapitola se dále zaměří na grafiku rastrovou. Je to způsob popisu (uložení, definice) zpracovávané informace formou rastrové matice (2D / 3D). Tento popis dat se získá: manuálně, syntézou (generováním nebo převodem z jiného popisu) nebo snímáním (kamerou atd.). Jednotkou matice je prvek zvaný pixel [11].

Pixely jsou definovány souřadnicemi a třemi hodnotami RGB (barevná složka). Nevýhodou rastrové grafiky je, že pokud objekt je jednou zobrazen v rastrové grafice, přestane jako takový existovat a dá se pracovat pouze s uloženou informací a to na úrovni pixelů. Rozlišení rastrové matice je dané dopředu a nelze jej jednoduše měnit, tak jak tomu je u grafiky vektorové. Převzorkování je jediná možnost jak měnit rozlišení matice, to se přitom buď zhorší (podvzorkování) nebo naopak zlepší (interpolace, vyhlazení)[11].

Počítačová 3D grafika využívá trojrozměrnou reprezentaci geometrických dat[8]. Pro realistický vjem scény v 3D grafice nestačí pouze tvary objektů a barvy, ale je třeba dodat textury, díky kterým je vjem více skutečný. 3D scéna, ve které je mnoho objektů pokrytých texturami o velkém rozlišení, by byla neúnosně paměťově náročná, proto je potřeba textury generovat. Jedním ze způsobů, jak generovat textury, je syntéza textur. Tato bakalářská práce se zabývá právě metodou jak syntetizovat textury rychle a bez větších ztrát na kvalitě.

2 Teorie

2.1 Co je to textura

Obecně slovo textura vyjadřuje vnitřní strukturu čí složení nějakého tělesa. Textura existuje v různých vědách a v každé z nich označuje něco odlišného. Například v geologii slovo textura znamená vlastnost hornin charakterizující prostorové uspořádání materiálů, v pedologii je pak vlastnost půd a zemin charakterizující zrnitostní složení, v typografii je tímto pojmem charakterizuje druh středověkého gotického písma[1,2].

Avšak tato publikace bude zaměřená na texturu z oblasti informačních technologií. Textura v počítačové grafice popisuje detailní struktury povrchu a materiálu objektu. Může charakterizovat různé povrchové vlastnosti, jako je terén, rostliny, minerály, kožešiny, kůži apod. Vzorek textury se nazývá texel [1,2].

2.2 Co je stochastický proces

V teorii pravděpodobnosti je stochastický proces, někdy také náhodný proces, protipólem procesu deterministického [8]. Ve stochastickém (náhodném) procesu jsou některé neurčitosti v jeho budoucím vývoji popsané rozdělením pravděpodobnosti. Dokonce to znamená, že když je počáteční podmínka počátečního bodu známá, je zde mnoho možností, jak by mohl proces běžet, ale některé cesty jsou více pravděpodobné než ostatní [8].

2.3 Texturování

Je to technika tzv. mapování textur, která nanáší textury na geometricky definovaný povrch objektu. Umožňuje dodat realistický vzhled virtuálnímu trojrozměrnému modelu. V podstatě to znamená určit barvu a jiné vlastnosti povrchu modelu a pomáhá tak rozpoznávat jednotlivé objekty a přibližovat se k realitě. Textury mohou mapovat různé vlastnosti [1,8]:

- **Barva povrchu** - nejčastější způsob použití textur, textura nanese na povrch objektu barevnou strukturu RGB.
- **Vlastnosti povrchu** – difůze a odrazivosti, mění charakter odrazu světla na povrchu objektu při výpočtu osvětlení. Simuluje lokálně matný nebo vyleštěný povrch, atd..
- **Světelné vlastnosti povrchu** - difůze a odrazivosti, mění charakter odrazu světla na povrchu objektu při výpočtu osvětlení. Simuluje lokálně matný nebo vyleštěný povrch, atd..
- **Průhlednost** - nerovnoměrné průhledné objekty, prolínání vlastností definovaných různými texturami přes sebe, oříznutí geometrie objektu nastavením úplné průhlednosti vybraným částem atd..
- **Modifikace normály** - Bump textury, provádí modifikaci normály podle hodnoty gradientu v textuře.
- **Změna geometrie** - displacement textury, provádí posun povrchových bodů ve směru normály podle hodnoty v textuře.

- **Hypertextura** - určuje optické vlastnosti nad povrchem objektu, vhodné pro zobrazení objektu typu ohně, vlasu, trávy apod..
- **Osvětlení** - light textury, realizují off-line difuzní statické osvětlení povrchu.

- **Zrcadlení okolí** - environment textury, nanáší na lesklý povrch obraz okolí.

Je možno nanášet na každý povrch více textur najednou (Multitexturing) [1,8].

2.3.1 Vytváření textur

Dle způsobu vytváření textur dělíme na obrázkové textury a procedurální textury. Zatímco u obrázkových je texturou předem připravený obrázek u procedurálních jsou textury vyjádřeny nějakou matematickou funkcí (vzorcem). U obrázkových je důležité rozlišení obrazu a tím i dostatečná detailnost textury, u procedurálních nezáleží na rozlišení, to se přizpůsobí velikosti renderovaného obrazu, avšak ne všechny povrchy se dají matematicky vyjádřit [1,8].

2.3.2 Mapování textur

Texturové mapování vyžaduje textury a u rozsáhlých objektů může požadovat značnou texturovou mapu. Platí to zvláště tehdy, když objekt je blízko pohledu. To znamená že, textura na povrchu je ve vysokém rozlišení, tudíž problémy s rozlišením texturové mapy vystoupí na povrch. Výsledky pokrytí texturových obrázků nemusí dopadnout dobře, protože v mnohých případech je obtížné získat obrázky, které jsou dobře pokryty – hranice musí být seřazeny a i když jsou, výsledná periodická struktura může být rozrušená. Záleží na zvoleném mapování a to především u obrázkových textur [1].

Je možné koupit obrázkové textury z různých zdrojů, ale bylo by idealní mít program, který umí generovat rozsáhlé texturové obrázky z malého vzorku a znázorňovat užitečnost reprezentovaných textur filtrováním výstupů [1]. Velmi důmyslné programy této formy je možné vytvořit.

2.3.3 Ukládání textur

Obrázkové textury se získávají úpravou digitálních fotografií (ořezáváním, filtrováním, apod) či kreslením pomocí nejrůznějších grafických editorů. Obrázkové textury jsou diskrétně nevzorkované a uloženy do paměti. Běžně používanými formáty pro textury jsou BMP, TGA a DDS (DirectDraw Surface) [1].

3 Syntéza textur

Od té doby co se počítačová grafika zabývá zejména reprodukcí skutečnosti do virtuální reality, textury se obecně používají při vytváření syntetických obrázků. Tyto textury lze získat z různých zdrojů např. ručně malovaných obrázků nebo digitálních fotografií. Obrázky pořízené fotoaparátem se realitě velmi přibližují, avšak často se stává, že nemají potřebnou velikost, to může vést k různým švům nebo opakování textury (viz obr.3.1), pokud používají přímo mapování textur. Aby k těmto jevům nedocházelo, používá tzv. syntézy textur [1,3].

Obrázek 3.1 – Opakováni textury
Zdroj [5]

Syntéza textur je alternativním způsobem tvorby textur. Snaží se vytvořit velký prostor z malého vzorku obrázků. Na základě vzorových obrázků se postaví pravděpodobnostní model textur a za pomocí jedné z metod syntézy textur se pokryje vzorky textur. Existuje mnoho metod jak sestavit takovýto model. Protože syntetické textury se mohou vytvořit v jakékoliv velikosti, předchází se visuálnímu opakování.

Její cíl může být stanoven následovně: Zadej vzorek textury, syntetizuj novou texturu, která, pokud ji vídí lidský pozorovatel, se zdá být generovaná stejným původním stochastickým postupem [1,3].

Hlavní problémy jsou[1]:

- **Modelování** - jak odhadnout stochastický proces z daného omezeného texturového vzorku.
- **Vzorkování** - jak vyvinout efektivní vzorkující proceduru k vytvoření nových textur z daného modelu.

Oba procesy modelování a vzorkování jsou důležité pro úspěšnou syntézu textury: visuální přesnost generovaných textur bude primárně záviset na přesnosti modelování, zatímco efektivita vzorkovací procedury bude přímo určovat výpočetní náklady generování textury [1].

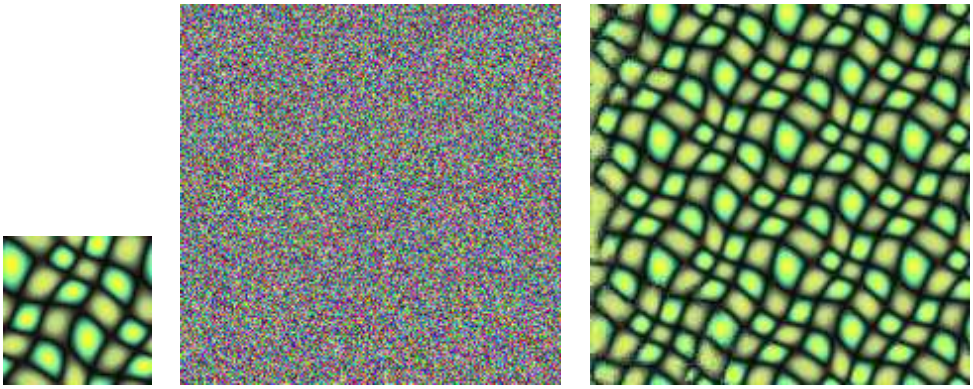
Vyobrazení modelů objektů jsou více realistická, když jsou modely texturovány (Má cenu přemýšlet nad tím, proč to může být pravda, i když bod je jasně daný). Existuje mnoho různých technik pro mapování textur. Základní myšlenkou je, že když je objekt vyobrazen, hodnota odrazivosti použitá ke stínování pixelu se získá odkazem na texturovou mapu. Nějaký systém souřadnic je přijat na povrchu objektu, aby spojil prvky texturové mapy s body na povrchu. Různé volby souřadnicového systému dávají vyobrazení, která vypadají zcela odlišně. Není vždy snadné zjistit, zda textura leží na povrchu přirozenou cestou (např. uvažujme vykreslení pruhů u zebry, kam by pruhy měly směřovat, aby vznikl přirozený vzor?). Kromě toho texturové mapování se zdá být důležitým trikem proto, aby vzniklé obrazové scény vypadaly realističtěji [1].

Hlavní strategií pro syntézu textur je přemýšlet o textuře jako o vzorku s nějakým rozdělením pravděpodobnosti a pak vyzkoušet a získat další vzorky použitím stejného rozdělení. Abychom učinili tento přístup praktickým, potřebujeme získát pravděpodobnostní model ze vzorku textury. Nejdříve je třeba předpokládat, že textura je homogenní (stejnorodá). To znamená, že lokální okna textury „vypadají stejně“ ať už byly vykresleny z kteréhokoliv místa v textuře. Rozdělení pravděpodobnosti hodnot pixelu je dáno vlastnostmi některého okolí tohoto pixelu spíše než pozicí tohoto pixelu [1,2].

Předpoklad homogenity znamená, že můžeme vytvořit model pro texturu mimo hranice oblasti našeho vzorku, založených na vlastnostech tohoto vzorku. Předpoklad často platí i na přírodní textury, které jsou nad rámec přiměřeného rozsahu stupnic. Například pruhy na zádech zebry jsou homogenní, ale je třeba si zapamatovat, že pruhy na zádech jsou vertikální a pruhy na jejích nohách jsou horizontální. Můžeme použít přikladu textury k získání pravděpodobnostního modelu pro syntetizovanou texturu různými způsoby [1,2].

V této kapitole je publikován jednoduchý algoritmus syntézy textur, který může efektivně syntetizovat širokou škálu textur, základní princip algoritmu je čerpán z materiálu [3]. Vstupem syntézy textur je vzorek textury a obrázek vyplněný náhodným šumem (Gaussovským šumem)[9], který má velikost odpovídající rozměrům výsledného syntetizovaného obrázku, které zadá sám uživatel jako vstupní parametry aplikace. V průběhu syntézy textur algoritmus modifikuje obrázek s náhodným šumem tak, aby vypadal jako vzorová textura [1,2].

Tato technika je flexibilní a snadná k použití, protože je zapotřebí pouze vzorku textury. Nové textury mohou být generovány v krátkém výpočetním čase a jejich podobnost vzorku je garantována, výsledkům syntézy bude věnována samostatná kapitola 5. Dvěma hlavními komponentami této metody syntézy textur jsou víceúrovňová pyramida (multiresolution pyramid) [8,3] a jednoduchý avšak výpočetně náročný prohledávací algoritmus [1,2].



(a) (b) (c)

Obrázek 3.2 - Vstupní a výstupní textury algoritmu Syntézy textur implementovaného v této bakalářské práci. Proces syntézy transformuje obrázek s vygenerovaným náhodným šumem(b) ze vzorové textury 64x64(a) na výslednou texturu 192x192(c). Zdroj vzorové textury (a) [5].

Hlavními výhodami této metody jsou rychlost a kvalita. Ve srovnání s ostatními současnými technikami je kvalita syntetizovaných textur stejná, dokonce v mnoha případech podává lepší výsledky. Výpočetní proces je o dva řády rychlejší [3] nežli u přístupů, které generují srovnatelné výsledky s tímto algoritmem. Proto se tento přístup dá využít v oblastech například syntéza pohybu či editace obrázků, kde je syntéza textur považována za velmi nákladnou (náročnou).

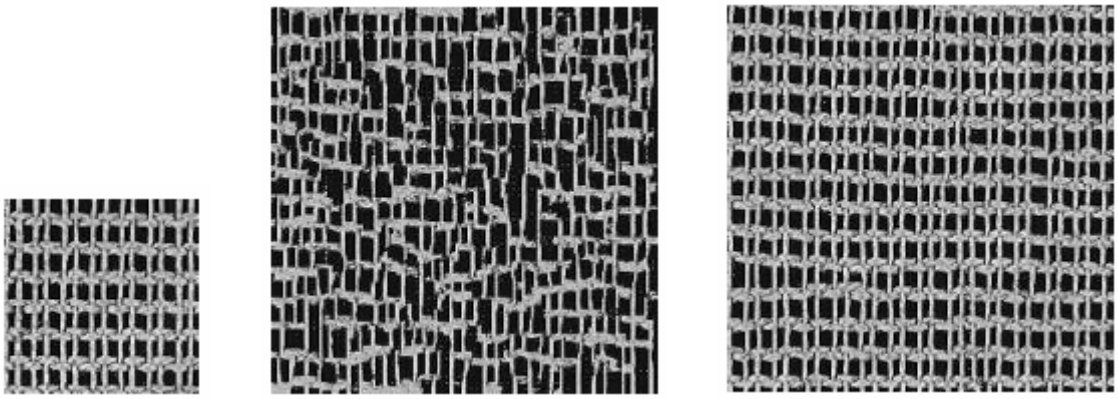
3.1 Současné přístupy k syntéze textur

Cílem této kapitoly není zde popisovat do detailů veškeré přístupy (algoritmy) k syntéze textur. Tato kapitola se zmíní pouze některé nedávno zpracované a významné práce. A především popíše metodu syntézy textur, ze které vychází přístup této bakalářské práce.

Fyzikální simulace: Určité povrchové textury je možné syntetizovat přímým simulováním jejich fyzikálních a vývojových procesů. Biologické předlohy jako kožešiny, kůže, šupiny mohou být modelovány za použití reakční difůze [8] a buněčného texturování [8]. Některé zvětrávající a minerální jevy mohou být úspěšně reprodukovány detailní simulací. Výhodou je, že tyto techniky mohou produkovat textury přímo v 3D síti a tak předchází problému s deformací při mapováni textur. Nevýhodou je, že fyzikální simulace je využitelná pouze na omezený třídy textur [3].

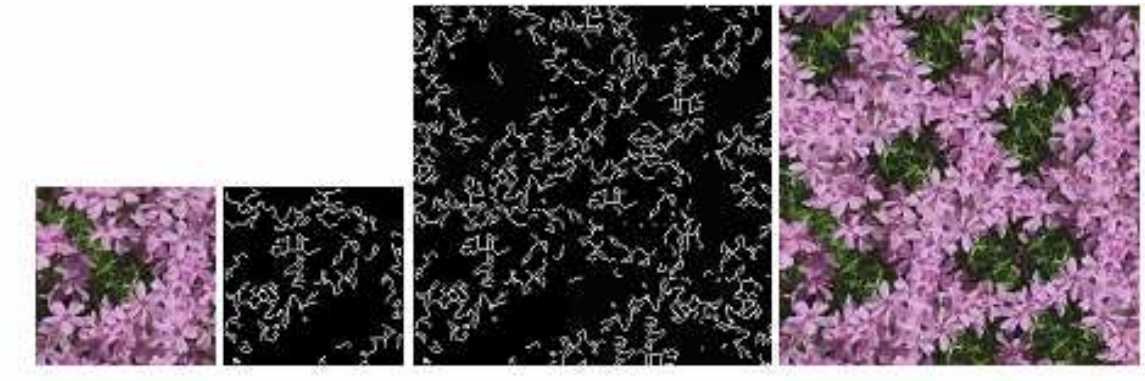
Markovovo náhodné pole (Markov Random Field) [4] a Gibbsovo vzorkování (Gibbs Sampling): Existuje mnoho algoritmu, které modelují textury pomocí Markovových náhodných polí nebo Gibbsova vzorkování. Výhodou je, že Markovou náhodná pole byla prokázána jako dobrá aproximace pro široký okruh textur, aproto jsou tyto algoritmy obecné a některé z nich dávají dobré výsledky (viz obr.3.3). Naproti tomu vzorkování za pomocí Markovova náhodného pole je velmi výpočetně náročné. Syntéza miniaturní textury může trvat klidně hodiny či dny [3].

Porovnání rysů (Feature Matching) 7]: Některé algoritmy modelují textury jako sadu rysů a vytvářejí nové obrázky na základě srovnání rysů ve vzorku textury (viz obr.3.4). Tyto algoritmy jsou obvykle účinnější než algoritmy Markovova náhodného pole. Heeger a Bergen modelovali textury porovnáváním hraničních histogramů obrázkových pyramid. Jejich technika byla úspěšná u vysoce náhodných (stochastických) textur, ale nebyla úspěšná u strukturovanějších textur. De Bonet syntetizoval nové obrázky transformací vstupního vzorku textury se zachováním závislostí cross-scale. Tato metoda pracuje lépe než [3] u strukturovaných textur, ale může produkovat okrajové artefakty, když vstupní textura není dobře pokryta. Simoncelli a Portilla vytvářejí textury porovnáním spojených statistik pyramid obrázku. Jejich metoda může úspěšně zachytit celkovou strukturu textury, ale není úspěšná v zachování lokálních vzorů [3].



(a) (b) (c)

Obrázek 3.3 - Metoda Markovovo náhodné pole (Markov Random Field) vzorové textury (a) generovala texturu s okolím 8 (b) a texturu s okolím 18 (c). Zdroj [4].



(a) (b) (c) (d)

Obrázek 3.4: Syntéza textur metodou Porovnání rysů (Feature Matching)[]pomocí map rysů (feature maps). Ze vzorku textury (a) se vytvoří mapa rysů (b), která se transformuje na mapu rysů o velikosti výsledné textury (c) a ta se transformuje na výslednou texturu (d). Zdroj[7].

Algoritmus přistupu této bakalářské práce je založen na algoritmu Fast Synthesis Texture (Li-Yi Wei a Marc Levoy) [3]. Metoda Wei a Levoye se snaží kombinovat předchozí přístupy, tak aby algoritmus efektivní, obecný a schopný vytvářet textury vysoce kvalitní a pokryté (viz obr.3.5). Měl by být také přátelský k uživateli, tj. počet vstupu by měl být minimální. Pro texturový model se používá Markovovo náhodné pole, jelikož tento přístup prokázal, že dokáže dobře syntetizovat široký okruh textur. Aby se předešlo výpočetní náročnosti Markovovy metody, Levoy a Wei [3] vyvinuli proceduru syntézy, která se vyhýbá pravděpodobným konstrukcím a vzorkování.

(a) (b) (c) (d)

Obrázek 3.5 - Vzorové textury (a)(c) a výsledné textury metody (b)(d) Li-Yi Wei a Marca Levoye. Zdroj[3].

Metoda Markovových náhodných polí modeluje texturu jako realizaci lokálních a statických náhodných procesů. To znamená, že každý pixel vzorové textury je charakterizován malou sadou prostorově sousedících pixelů (okolí pixelu). Tato charakteristika je u všech pixelů stejná. Lze si představit, že se pozorovateli dá obrázek, ale může ho vidět pouze malým pohyblivým okýnkem. Jak se okno pohybuje, pozorovatel může vidět různé části obrázku. Obrázek je statický, když při určité definované velikosti okna je pozorovaná část vždy podobná. Obrázek je lokální, když se každý pixel dá předpovědět z malé sady sousedících pixelů (okolí) a je nezávislý na zbytku obrázku [3].

Za předpokladu těchto lokalit a stacionarit algoritmus syntetizuje novou texturu tak, že je lokálně podobná příkladu texturového pole. Nová textura vzniká pixel po pixelu a každý pixel je určen tak, že lokální podobnost mezi vzorkem textury a výsledným obrázkem se uchovává. Tato procedura je zcela deterministická, i když je založena na metodě Markovových náhodných polí a

žádné explicitní rozdělení pravděpodobnosti není zkonstruováno. Výsledkem je efektivní algoritmus, který lze dál akcelerovat [3].

3.2 Algoritmus

Jak bylo již zmíněno algoritmus je založen na přístupu Wei a Levoye[3], využívá Markovových náhodných polí jako texturového modelu. Cílem této metody je vytvořit novou texturu tak, že každá její lokální oblast je podobná jiné oblasti ze vstupní textury. V následující kapitole 3.2.1 bude popsano jednoduché řešení, v další kapitole 3.2.2 bude popsáno řešení pomocí Gaussovy pyramidy, které zefektivní výsledek syntézy.

3.2.1 Jednoúrovňová varianta

Algoritmus začíná se vstupním vzorkem textury $\boldsymbol{I_a}$ a obrázkem s Gaussovým náhodným šumem $\boldsymbol{I_s}$, který je potřeba si vygenerovat[9]. Algoritmus se snaži transformovat obrázek s Gaussovým šumem, tak aby vypadal jako vzorová textura $\boldsymbol{I_a}$. Syntéza probíhá transformací $\boldsymbol{I_s}$ pixel po pixelu v rastrovém uspořádání. K určení hodnoty pixelu $\boldsymbol{p}$ u $\boldsymbol{I_s}$ je nutné srovnat jeho prostorové okolí $\boldsymbol{N(p)}$ se všemi možnými okolími $\boldsymbol{N(p_i)}$ vzorové textury $\boldsymbol{I_a}$. Vstupní pixel $\boldsymbol{p_i}$ s nejpodobnějším $\boldsymbol{N(p_i)}$ je označen jako $\boldsymbol{p}$ [3]. K měření podobnosti $\boldsymbol{P}$ mezi okolími je použita rovnice 3.1.

$$P = \sqrt{\sum_{i=0}^{n}(S_{R_i} - T_{R_i})^2} + \sqrt{\sum_{i=0}^{n}(S_{G_i} - T_{G_i})^2} + \sqrt{\sum_{i=0}^{n}(S_{B_i} - T_{B_i})^2} \quad (3.1)$$

,kde $\boldsymbol{n}$ je počet pixelů v okolí (v masce), $\boldsymbol{S_{Rn}}$, $\boldsymbol{S_{Gn}}$ a $\boldsymbol{S_{Bn}}$ jsou hodnoty RGB pixelu textury s náhodným šumem, $\boldsymbol{T_{Rn}}$, $\boldsymbol{T_{Gn}}$ a $\boldsymbol{T_{Bn}}$ jsou hodnoty RGB pixelu vzorové textury. Poté stačí vybrat minimum ze všech podobností pixelů vzorové textury a určit tak pixel. Cílem procesu syntézy je zajistit, že nově označený pixel **p** zachová co nejvíce lokální podobnosti mezi okolími $\boldsymbol{I_a}$ a $\boldsymbol{I_s}$, jak je možné. Tentýž proces se opakuje u každého vstupního pixelu, až se určí všechny pixely. Je to podobné jako, když se skládá puzzle: kousky jsou individuální pixely a vhodnost mezi těmito kousky je určená barvou okolních sousedních pixelů [3].

Protože se sada lokálních okolí $\boldsymbol{N(p_i)}$ používá jako primární model pro textury, kvalita výsledku bude záviset na jejich rozlišení a tvaru. Pravděpodobně by měla být velikost okolí v poměru s největší pravidelnou strukturou textury, jinak se struktura může změnit a potom bude výsledný obrázek vypadat příliš náhodně [3].

Tvar okolí bude přímo určovat kvalitní $\boldsymbol{I_s}$. Okolí může obsahovat jen ty pixely, které předchází aktuálnímu výstupnímu pixelu v rastrovém uspořádání [3].

3.2.2 Víceúrovňová varianta

Algoritmus jednoduchého řešení zachycuje texturové struktury s použitím adekvátně velkých okolí. Pro textury obsahující stupnicovité struktury (složité struktury složené z více struktur jednodušších) se však musí použít rozsáhlá okolí a tyto okolí vyžadují více výpočtů. Tento problém je možné vyřešit použitím víceúrovňové pyramidy (multiresolution pyramide) [2,3]. Protože je možné reprezentovat

stupnicovité struktury kompaktněji pomocí mála pixelů na určité úrovni pyramidy s nižším rozlišením, klesne výpočetní náročnost algoritmu [3].

Algoritmus víceúrovňové syntézy pokračuje následovně. Nejdříve se sestaví dvě Gaussovy pyramidy G_a z I_a a G_s z I_s nezávisle na sobě. Pyramida se vytváří tak, že se jako úroveň 0 pyramidy stanoví obrázek (vzorek textury nebo obrázek s náhodným šumem), následující úroveň se určí tak, že je obrázek filtrován Gaussovým filtrem a podsamplován na polovinu. Takto se postupuje, dokud jsou rozměry okolí (masky) menší než rozměry úrovně. Algoritmus pak postupně transformuje pixely G_s od stupně s nejnižším rozlišením do stupně s maximálním rozlišením tak, že každá úroveň G_s s vyšším rozlišením se konstruuje z již syntetizovaných úrovní s nižším rozlišením. Je to podobné postupu, jakým se maluje obraz: nejdříve malíř nakreslí dlouhé a silné čáry (tahy) a postupně kreslí detailněji. V průběhu každé výstupní úrovně pyramidy $G_s(L)$ se pixely syntetizují podobným způsobem jako v případě jednoúrovňové varianty. Jedinou změnou pro víceúrovňový přístup je, že každé okolí $N(p)$ obsahuje pixely v aktuálním rozlišení stejně jako, ty v rozlišení nižším. Podobnost mezi dvěma víceúrovňovými okolími se měří vypočtením sumy druhé mocniny rozdílu všech pixelů v nich (okolích). Pro všechny úrovně pyramidy mimo vrcholné úrovně, platí rovnice 3.2. Pro nejvyšší vrstvu platí rovnice 3.1.

$$P = \sqrt{\sum_{i=0}^{n}(S(L)_{R_i} - T(L)_{R_i})^2 + \sum_{j=0}^{m}(S(L-1)_{R_j} - T(L-1)_{R_j})^2} + \sqrt{\sum_{i=0}^{n}(S(L)_{G_i} - T(L)_{G_i})^2 + \sum_{j=0}^{m}(S(L-1)_{G_j} - T(L-1)_{G_j})^2} + \sqrt{\sum_{i=0}^{n}(S(L)_{B_i} - T(L)_{B_i})^2 + \sum_{j=0}^{m}(S(L-1)_{B_j} - T(L-1)_{B_j})^2} \quad (3.2)$$

, kde m je počet pixelů okolí předchozí úrovně (druhé masky) a L je level pyramidy.

3.2.3 Manipulace s okraji

Správná zacházení s okraji vzorku textury je velmi důležité. U pyramidy syntézy se okraj zvětšuje prstencově. Často nastává problém, že několik pixelů okolí $N(p_i)$ sahá za okraje vzorové textury. To je vyřešeno tak, že když $G_s(L,x,y)$ označuje pixel na úrovni L v pozici (x,y) pyramidy G_s, pak [3]:

$$G_s(L,x,y) = G_s(L,x \bmod M, y \bmod N) \quad (3.3)$$

,kde M a N jsou čísla řad a sloupců $G_s(L)$. Takováto prstencová manipulace s okraji je základem garance, že se na výsledné textuře nebudou objevovat švy.

Pro vstupní pyramidu $\mathbf{G_a}$ prstencová okolí typicky obsahují nesouvislosti (trhliny), pokud není I_a pokryté. Řešení spočívá v použití pouze těch pixelů $N(P_i)$, které jsou zcela uvnitř G_a a vyřazení těch, které přesahují okraje [3].

3.2.4 Souhrn algoritmu

Průběh mého přístupu k syntéze textur je popsán v následujícím pseudokódu. Pro jednoduchost funkce (metoda) `synthesis(Ia)` má jeden parametr a to vstupní texturu I_a, ovšem ve skutečnosti obsahuje daleko více parametrů jako např. velikost okolí (maska), volba urychlení apod. Funkce `VytvorPyramidy(Ia, Is)`, představuje ve skutečnosti třídu pyramide, která vytvoří pyramidu G_a a G_s. Následuje výpočetně náročný algoritmus, který prochází úrovně pyramid od vrcholu k základně, porovnává mezi sebou vrstvy pyramidy G_s a G_a úrovně L a výsledek předchozí úrovně $L+1$. Funkce urciMinimum pouze nahrazuje část algoritmu, která vybírá nejpodobnější pixel z G_a, jímž se nahradí aktuální pixel v G_s. Pro snadnější pochopení nejsou v pseudokódu zahrnuty aplikované algoritmy pro urychlení, ty popisuje kapitola 3.3.

```
Function synthesis (Ia)
    Ga,Gs  ← VytvorPyramidy(Ia, Is);
    foreach level L od nizsich k vyssim rozlisenim Ga (od n k 0)
       for (vsechny pixely (xs, ys) pyramidy Gs)
          for (vsechny pixely (xa, ya) pyramidy Ga)
             // urci rozdily okoli Ga a Gs v L (r–red.g–green,b–blue)
             for (všechny pixely okoli L)
                sumaRed += (r(Gs[L][xs][ys]) – r(Ga[L][xa][ya]))^2;
                sumaGreen += (g(Gs[L][xs][ys]) – g(Ga[L][xa][ya]))^2;
                sumaBlue += (b(Gs[L][xs][ys]) – b(Ga[L][xa][ya]))^2;
             if (level L není vrcholem pyramidy)
                // urci rozdily okoli Ga  a Gs  predchoziho levelu L+1
                for (všechny pixely okoli L+1)
                   sumaRed += (r(Gs[L+1][xs][ys]) – r(Ga[L+1][xa][ya]))^2;
                   sumaGreen += (g(Gs[L+1][xs][ys]) – g(Ga[L+1][xa][ya]))^2;
                   sumaBlue += (b(Gs[L+1][xs][ys]) – b(Ga[L+1][xa][ya]))^2;
             // urci vektor slozek RGB, ktery urci nejvhodnejsi pixel z Ga pro Gs
             vector = √(sumaRed)+ √(sumaGreen)+ √(sumaBlue);
             min ← urciMinimum(vector);
          Gs[L][xs][ys] ← min;
    return Gs[0];
```

Algoritmus 3.1 – pseudokód popisuje průběh metody této bakalářské práce

3.3 Akcelerace

Syntéza textur je vysoce výpočetně náročná. Metoda aplikovaná v této bakalářské práci je sice rychlejší než předešlé techniky, ale proces syntézy větších obrázků při větším okolí je zdlouhavý, trvá minuty nebo i desítky minut. Výpočetní jednotka je velmi zatížená a výpočet není rychlý. Proto je výhodné použít nějaké urychlení.

Rychlost algoritmu ovlivňují následující faktory:

1. Programovací jazyk a druh aplikace
2. Přístup k datům
3. Dílčí urychlení klasické metody
4. Metody urychlení

3.3.1 Volba programovacího jazyka a typu aplikace

Rychlost algoritmu je závislá na volbě programovacího jazyka a typu aplikace. Má aplikace využívá .NET Microsoft Windows. Algoritmus jsem implementoval 3 způsoby:

- Konzolová aplikace C++ s využitím knihovny CImg
- Windows Forms aplikace C++ s využitím knihovny CImg
- Windows Forms aplikace C# (pro práci s obrázky bylo využito System::Drawing)

Jednoznačně se prokázalo, že nejrychlejší výpočet dává konzolová aplikace C++. Jako celkem použitelná se osvědčila Windows Forms aplikace C++. Zatímco naprosto nepoužitelná byla Windows Forms aplikace C#. To vypovídá o tom, že programování výpočetně náročné aplikace v jazyku C# je méně vhodné než v C++ . A také o tom, že aplikace typu Windows Forms (události, procesy) zpomaluje průběh algoritmu.

3.3.2 Přístup k datům

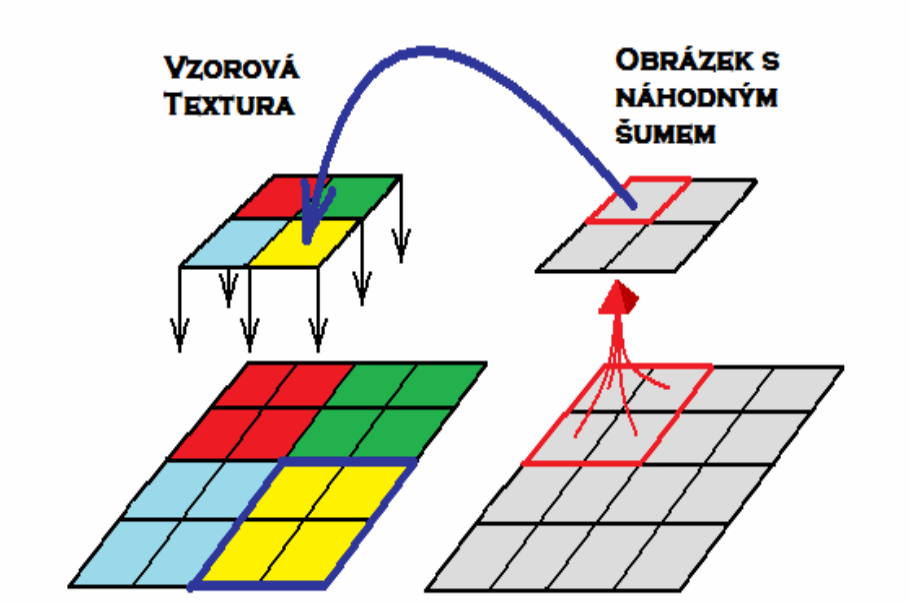
Dalším faktorem ovlivňujícím urychlení výpočtu je volba přístupu k datům. V této aplikaci jsem zkoušel implementovat jak přímý přístup k hodnotám RGB obrázku CImg či Systém::Drawing::Bitmap (práce přímo s hodnotami obrázku), tak přístup k těmto hodnotám přes matice (uložení hodnot obrázku do matice, práce s maticí, zpětné uložení hodnot matice do obrázku). Ukázalo se, že v aplikaci C# přístup přes matice byl mnohem rychlejší než přístup přímý. Naopak v C++ se projevila přímá práce s hodnotami obrázku rychlejší.

3.3.3 Dílčí urychlení klasické metody

Algoritmus je možno upravit tak, aby využíval již zpracovaných výsledků. Například při přechodu na okolí následujícího bodu se zaznamená mezivýsledek z bodů okolí, které se budou opakovat a pak stačí jen přičíst hodnotu bodů nových. Pro složitost algoritmu a faktu, že výpočet se řádově nezrychlí, jsem této metody nevyužil v bakalářské práci.

3.3.4 Metoda FW-PLI (Find Way by Previous Level Information)

Metoda, jež sem autorem, je založena na principu stromu. Postup vypadá následovně: Při syntéze vyšší úrovně se pro každý bod $G_s(x, y)$ obrázku s náhodným šumem uloží souřadnice nejpodobnějšího bodu vzorové textury. V následující úrovni se vzorová textura rozdělí na 4 kvadranty. Nyní se využijí souřadnice bodu vzorku textury, který je nejpodobnější bodu G_s, k volbě kvadrantu, v němž se pro body $G_s(x,y)$, $G_s(x,y+1)$, $G_s(x+1,y)$ a $G_s(x+1,y+1)$ budou vyhledávat pixely nejvíce podobné. Urychlení spočívá v prohledávání pouze čtvrtiny (jednoho kvadrantu) každé vrstvy pyramidy vzorové textury. Průběh metody znázorňuje obrázek 3.6.

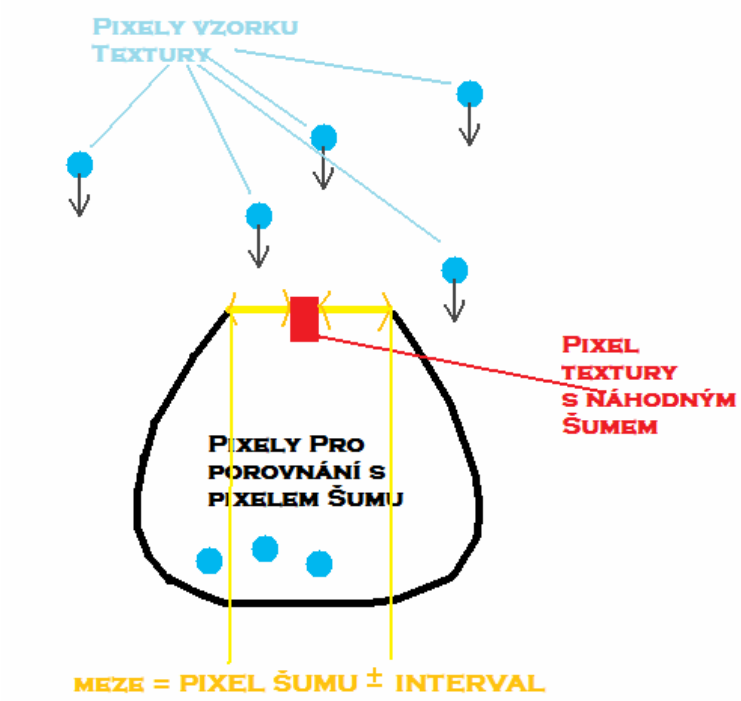


Obrázek 3.6: Metoda urychlení FW-PLI, napravo je pyramida šumu, nalevo pyramida vzorové textury. Obrázek znázorňuje průběh metody Find Way By Previous Level Information.

3.3.5 Metody pro omezení (metoda „Omezení intervalem“)

Pro multidimenzionální vektor textury náhodného šumu se hledá z množiny multidimenzionálních vektorů vzorové textury ten, který je mu nejpodobnější. Klasická metoda bez urychlení prochází všechny vektory množiny a najde optimální řešení. Každý dílčí výpočet je unikátní a nelze jej zjednodušit. Proto je snahou najít nějaký užší výběr z množiny vektorů (eliminovat vektory, pro které lze předpokládat, že v nich hledané minimum nenastane. Proto existuje řada metod například TSVQ (Tree-structured Vector Quantization)[], které logaritmicky sníží dobu výpočtu. TSVQ využívá algoritmus Fast Synthesis Texture (Li-Yi Wei a Marc Levoy) [3]. Metody spočívají v tom, že připraví množinu vektorů, pro každý vektor určí klíč a sestaví vektory do binárního stromu. Hledání nejbližšího vektoru pak spočívá v průchodu tohoto stromu podle stanoveného klíče. Řada odborných materiálů se zmiňuje o těchto metodách, avšak nepopisuje způsob stanovení tohoto klíče, což je pro úspěšnost metody podstatné.

V bakalářské práci jsem vymyslel a implementoval jednoduchou metodu „Omezení intervalem“ pro zmenšení množiny prohledávaných vektorů. Předem vytvořím 3 matice spočítaných velikostí vektorů (pro složku R, G a B zvlášť) pro všechny body textury (vektor je složen hodnotami RGB všech bodů okolí dle určené masky). Při průchodu všemi body zpracovávaného obrázku šumu pro každý bod spočítám z jeho okolí obdobně velikost jeho vektorů pro R, G a B. Do zpracovávané množiny bodů z textury použiju jen ty, jejichž všechny tři složky RGB spadají do intervalu <***složkaRvektoru – zadanaMez, slozkaRvektoru + zadanaMez***>. To nasvědčuje tomu, že čím je hodnota ***zadanaMez*** větší, tím bude výsledek lepší, avšak výpočet delší (viz obr.3.7). Vhodnou volbou hodnoty ***zadanaMez*** tak mohu stanovit optimální způsob výpočtu pro zadané podmínky.



Obrázek 3.7 - Modré balónky představují pixely vzorové textury, které pokud se vejdou všemi třemi hodnotami R,G a B do mezí, tak se budou porovnávat s pixelem textury náhodného šumu.

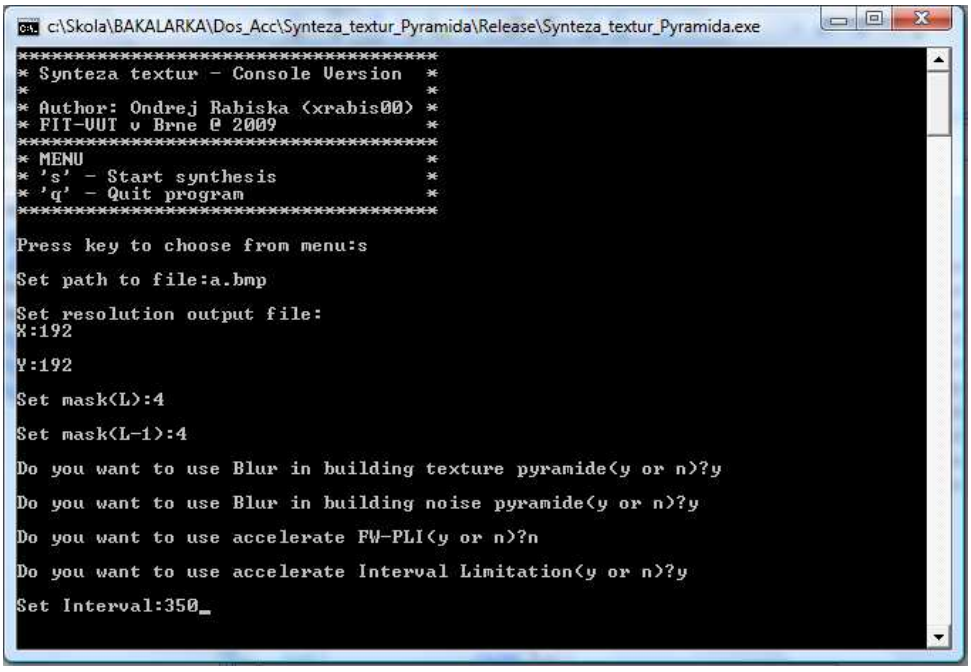
4 Aplikace

Nejvhodnějším programovacím jazykem pro tvorbu této aplikace bylo C++, protože relativně rychle zpracovává výpočetně náročný algoritmus implementovaný v této bakalářské práci. Aplikace v C++ .Net Windows Forms, zpracovávala algoritmus asi 3,5 krát pomaleji než konzolová aplikace, avšak protože prostředí windows forms je uživatelsky přátelské, zahrnul jsem obě varianty do tohoto projektu.

4.1 Konzolová aplikace

Program v C++ využívá pro načtení a prácí s obrázky knihovny CImg [10]. Zahrnuje jednoduché menu, kde jsou na výběr dvě možnosti Start a Quit.

Ovládání programu je velmi jednoduché: Nejdříve výběrem **Start synthesis (‘s‘)** spustíme syntézu textur, ještě před tím než se spustí složitý a náročný algoritmus je třeba nastavit jeho parametry: cestu k vstupní vzorové textuře (**Set path to file**), rozlišení výstupní textury (**Set resolution output file**), masku úrovně **L** (**Set mask(L)**), masku úrovně předešlé k **L** (Set **mask(L-1)**), aktivaci Gaussova filtru (blur) při stavbě pyramidy vzorové textury (**Do you want to use Blur in building texture pyramide (y or n)?**), aktivaci Gaussova filtru (blur) při stavbě pyramidy obrázku s náhodně generovaným šumem (**Do you want to use Blur in building noise pyramide(y or n)?**) a jako poslední parametry zrychlení metodou FW-PLI (**Do you want to use accelerate FW-PLI(y or n)?**), u které je třeba nastavit interval (**Set Interval**). Poté proběhne syntéza a po jejím skončení, se program vrátí do menu, kde volbou **Quit program (‘q’)** skončí (viz.obr.4.1).



Obrázek 4.1 - Vzhled a ovládání konzolové aplikace Syntéza textur

4.2 Aplikace .NET Windows Forms C++

Uživatelsky přátelštější aplikace Windows Forms v C++ využívá taktéž knihovny CImg pro zpracování obrázků. Pro načítání a ukládání obrázků používá Systém::Drawing. Program se skládá ze dvou základních částí hlavní program (formulář) a proces syntézy, pro nějž bylo vytvořeno nové vlákno (thread). Aplikace obsahuje dva formuláře, prvním z nich je hlavní okno, ve kterém se nastavují parametry syntézy, progress bar ukazuje průběh syntézy textur, na levé straně je okno, které zobrazuje otevřenou vzorovou texturu, krerá se bude syntetizovat, pod ním v Location se vypíše cesta k této textuře a napravo od zobrazovacího okna jsou parametry rozlišení textury. Uprostřed je panel Set Attributes, ve kterém se nastavují parametry syntézy textur. Vpravo je okno, které po dokončení procesu syntézy zobrazí výsledek, pod ním jsou parametry rozlišení výstupního obrázku a možnosti jako uložit výsledek (Save) a porovnat výsledek s ostatními obrázky Compare Images. Dole v tomto formuláři je panel Status obsahující informace o průběhu syntézy např. progressbar. Druhý formulář se zobrazí po kliknutí na Compare Images a slouží pouze jako doplněk aplikace, uživatel si v něm může výsledný obrázek porovnávat se dvěma dalšími. Ovládání je snadné a popisuje ho tutoriál v příloze B.

5 Výsledky

Obsahem této kapitoly je prezentace výsledků testování algoritmu aplikovaného v této bakalářské práci. Při testování byly použity nejrůznější textury. Budou se porovnávat obrázky vygenerované syntézou textur s vzorovou texturou s nejrůznějšími parametry a použitými urychlovacími metodami.

Při testování syntézy textur byla sledována dvě hlavní kriteria – doba výpočtu a kvalita výsledného obrázku. V dalších kapitolách je uvedeno, co uvedená kriteria ovlivňuje a jak.

5.1 Doba výpočtu

Doba výpočtu závisí na

- vstupních parametrech, kterými jsou:
  - použitá maska
  - rozměry vzorové textury
  - rozměry výsledné textury
  - výška pyramidy
- typu aplikace
  - C++ konzolová aplikace
  - C++ Windows aplikace
  - C# aplikace
- zvoleném algoritmu (a jeho urychlovací metody)
  - klasická metoda bez urychlení
  - metoda FW-PLI
  - metoda IL s různým intervalem

Doba výpočtu samozřejmě také závisí na HW, na kterém je výpočet realizován.

Výsledky testování zaměřené na dobu výpočtu je možno vidět v tabulce 5.1. Testování proběhlo na jednom vzorku textury a na stejném HW, aby měření bylo objektivní. Po srovnání jednotlivých časů lze říci, v použitém algoritmu mají urychlovací metody své opodstatnění.

Urychlovací metoda		bez urychlení	FW-PLI	IL(150)	IL(200)	IL(250)	IL(300)	IL(350)	FW-PLI IL(330)
Maska 2	Console App	1:26.79	0:22.63	0:16.56	0:29.51	0:42.72	0:54.60	1:05.20	0:15.37
	WinForms App	5:26.79	1:31.89	1:04.14	1:57.14	2:38.01	3:25.71	4:06.25	1:02.86
Maska 3	Console App	2:27.23	0:37.40	0:19.39	0:38.88	0:58.47	1:17.02	1:32.87	0:22.26
	WinForms App	8:58.58	2:27.68	1:14.80	2:26.33	3:45.95	4:50.86	5:44.06	1:28.64
Maska 4	Console App	3:45.61	0:57.68	0:22.47	0:51.41	1:22.14	1:49.07	2:11.15	0:31.97
	WinForms App	13:56.01	3:42.02	1:24.82	3:15.55	5:06.67	6:43.10	8:06.74	2:05.66

Tabulka 5.1 - V tabulce jsou výsledky časů výpočtů aplikované syntézy textur jak konzolové aplikace, tak aplikace .NET Windows Forms. Byl testován algoritmus bez urychlení, s akcelerací FW-PLI, s metodou „Omezení intervalu“pro intervaly 150, 200, 300, 350, v poslední řadě algoritmus s kombinací obou urychlujících metod. Výsledky byly naměřeny s parametry: vstupním vzorkem textury o velikosti 64 x 64, obrázkem náhodné textury s rozlišením 192 x 192, s maskami 2,3,4.

V dalším textu je podrobněji rozepsáno, jak jednotlivá dílčí kriteria ovlivňují rychlost výpočtu.

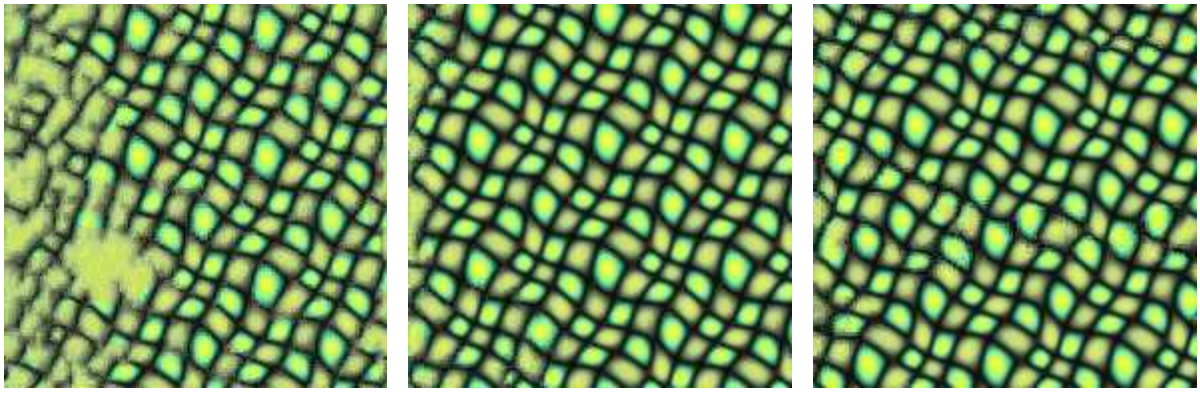
5.1.1 Kriterium pro rychlost výpočtu - velikost masky

Ovlivňuje čas a kvalitu výsledku. Velikost masky představuje velikost okolí zkoumaného pixelu zahrnuté do porovnávacího algoritmu. Následující příklad ukazuje rozdíly při použití různých velikostí masek (2, 3, 4). Optimálním řešením v tomto případě je použít masku 3. Znamená to, že použití větší masky nemusí vést ke kvalitnějšímu výsledku.

Maska 2 - výsledky časů výpočtu syntézy textur s maskou 2 jsou sice nadmíru uspokojivé, avšak výsledná textura již tak ohromující není. Některé části textury jsou rozrušené a obsahují švy. Proto tyto oblasti nejsou příliš podobné vzorové textuře (viz Obrázek 5.1 a).

Maska 3 – ve srovnání s maskou 2 jsou časy průběhu algoritmu o něco větší, avšak výsledná textura je téměř totožná se vzorkem textury. Jen málo se objevují švy a rozrušení textury (viz Obrázek 5.1 b).

Maska 4 - výsledky testování procesu syntézy s maskou 4 jsou nepříliš uspokojivé, výsledné časy jsou pochopitelně vyšší než u masky 3, avšak výsledek je horší než u masky 3 (viz Obrázek 5.1 c).



(a) *(b)* *(c)*

Obrázek 5.1 - Na obrázku je syntetizovaná textura metodou bez urychlení s velikostí masky 2 (okolí 5x3 a 3x3), jsou zde patrné trhliny a švy. Textura s maskou 3 (okolí 7x4 a 4x4) téměř odpovídá vzorové textuře. Obrázek s maskou 4 (okolí 9x5 a 5x5) je také uspokojivá, avšak není tak pravidelná.

5.1.2 Kriterium pro dobu výpočtu – typ aplikace

Na rychlost výpočtu má vliv i typ aplikace a jazyk, ve kterém byla vytvořena. V této bakalářské práci byly vytvořeny tři typy aplikací řešící daný problém – konzolová aplikace C++, Windows Forms aplikace C++ a Windows Forms aplikace C#.

Ukázalo se, že nejlepší výsledky z hlediska času dává konzolová aplikace C++, která je ovšem méně uživatelsky přívětivá.

Výpočet s Windows Forms aplikací C++ byl také přijatelný. Použití jazyka C++ umožnilo využít pro výpočet knihovnu <CIMG> a odpovídající formát obrázku, což se ukázalo jako podstatně rychlejší než použití třídy bitmap v aplikaci C#. Tam sice převedení bitmapy do matice podstatně urychlilo výpočet, ale výsledek jako celek nebyl uspokojivý.

5.1.3 Kriterium pro doby výpočtu - použitý algoritmus

Metoda Find Way by Previous Level Information je nejrychlejší, pokud metody nekombinujeme. Protože využívá informací z předchozích úrovní, její výpočetní náročnost klesla 3-4x. Jejím optimálním řešením je vygenerovaný obrázek s maskou 3, který je vzorovému obrázku velmi podobný, avšak jeho struktura je odlišná nepříliš hierarchická. Výpočet je 4x rychlejší než u algoritmu bez urychlení.

Metoda Interval Limition (LI) je sice při malém rozmezí (například 150) rychlejší, ale výsledky její syntézy jsou mizerné. Podává velmi dobré výsledky, pokud se rozmezí pohybuje ± 300. Tato metoda s intervalem 350 je přibližně 1,6x rychlejší než metoda klasická bez urychlení.

Metody kombinovaná využívající, jak urychlení pomocí FW-PLI, tak pomocí „Omezení intervalem (při rozmezí kolem ±330) je 1,6x rychlejší než samotné FW-PLI a 6,7x rychlejší než metoda klasická bez jakéhokoliv urychlení, tím se stala nejrychlejším postupem podávající kvalitní výsledky.

Klasická metoda bez urychlení je sice zdlouhavá, avšak podává nejlepší výsledky.

5.2 Kvalita výsledku

Neméně důležitým kriteriem a předmětem testování je kvalita výsledné textury. Ta závisí na

- velikosti masky
- výšce pyramidy, -
- složitosti vstupní textury
- zvoleném algoritmu syntézy textur (a jejich parametrech)
- použití parametrů texture Blur a Noise Blur
-

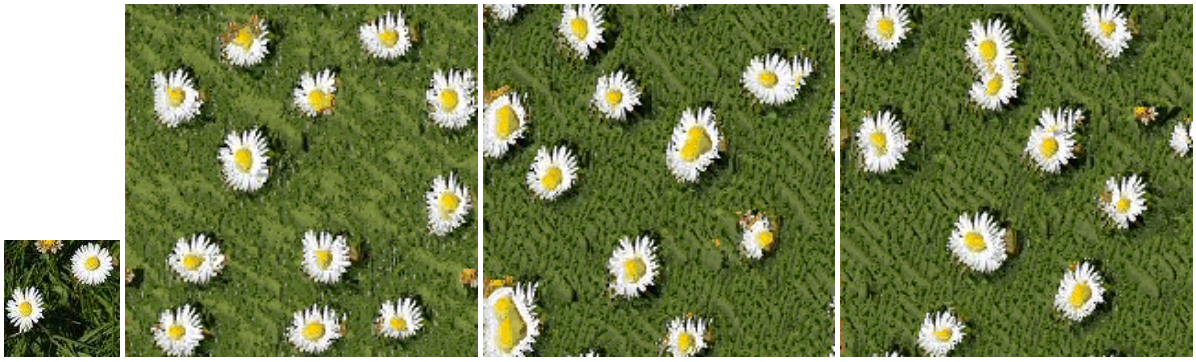
5.2.1 Kriterium kvality – použití rozmazání textury

Parametry Texture Blur a Noise Blur znamenají, že při tvorbě vyšší vrstvy pyramid budou vzorové textury a obrázky se šumem před zmenšením vrstvy textury rozmazány

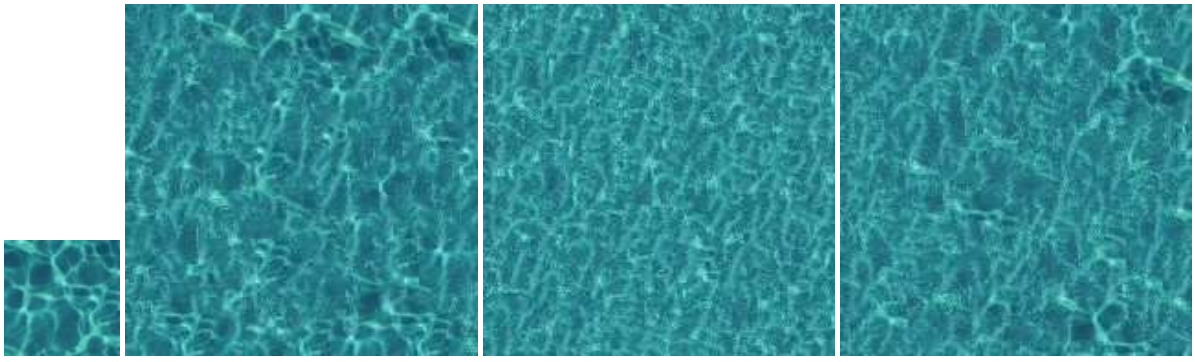
Metoda s aktivními parametry rozmazání vytváří různé výsledky a nezbývá nic jiného než, aby člověk posoudil sám, který z výsledků je lepší (viz obr.5.2, obr.5.3, obr.5.4).

(a) (b) (c) (d)

Obrázek 5.2 - Realistický vzorek textury „kamení“. (a) vzorový obrázek, (b) výsledek bez aplikování Gaussových filtrů, (c) aplikace Gaussova filtru na úrovně pyramidy vzorové textury (Texture Blur)a (d) aplikace Gaussova filtru na úrovně jak vzorové textury tak textury s náhodným šumem (Noise Blur). *Zdroj (a) je [6]*.



Obrázek 5.3 - Realistický vzorek textury „tráva se sedmikráskami“. (a) vzorový obrázek, (b) výsledek bez aplikování Gaussových filtrů, (c) aplikace Gaussova filtru na úrovně pyramidy vzorové textury (Texture Blur)a (d) aplikace Gaussova filtru na úrovně jak vzorové textury tak textury s náhodným šumem (Noise Blur). Zdroj (a) je [6].

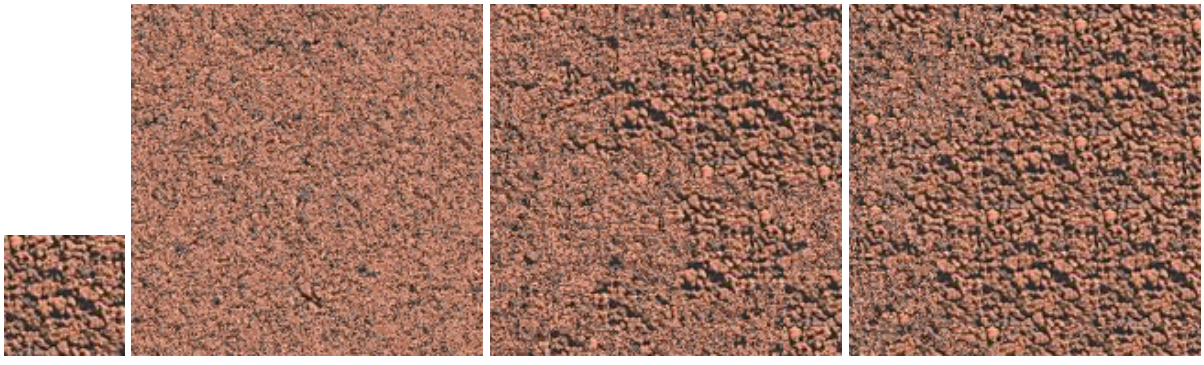


Obrázek 5.4 - Realistický vzorek textury „voda“. (a) vzorový obrázek, (b) výsledek bez aplikování Gaussových filtrů, (c) aplikace Gaussova filtru na úrovně pyramidy vzorové textury (Texture Blur)a (d) aplikace Gaussova filtru na úrovně jak vzorové textury tak textury s náhodným šumem (Noise Blur). Zdroj (a) je [5].

5.2.2 Kriterium kvality - velikost masky

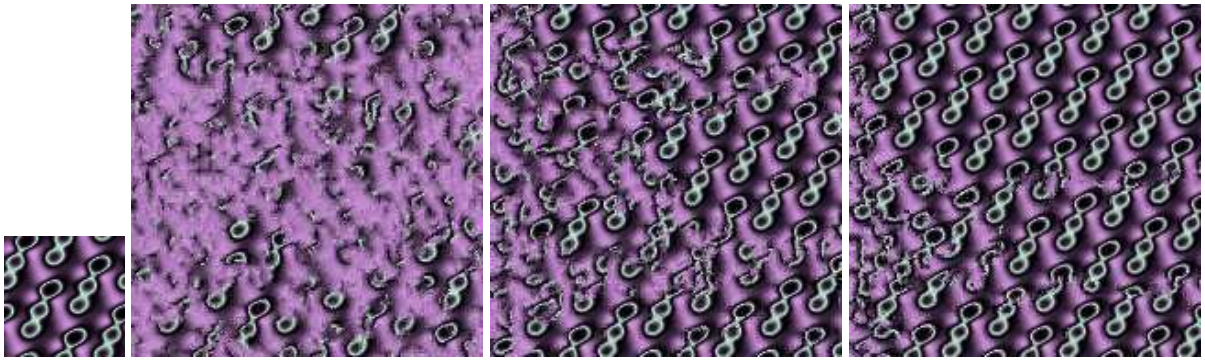
Jak ovlivňuje velikost masky rychlost a kvalitu výpočtu již bylo ukázáno v předchozí podkapitole.

Obecně bylo zjištěno, že pokud má vzorová textura jednoduchou nečlenitou strukturu, která se hodně opakuje stačí maska malá (viz obr.5.5). Pokud je struktura vzorku textury členitější je potřeba maska větší, ale ne příliš velká, protože potom by docházelo ke švům (viz obr.5.6).



(a) (b) (c) (d)

Obrázek 5.5 - Klasický přístup bez urychlení, vstupní textura (a) o rozměrech 64x64 je málo strukturovaná a velmi často se opakuje. Na následujících obrázcích jsou výsledky syntézy textur s maskou 2(b), 3(c), 4(d). Zdroj (a) je [5].

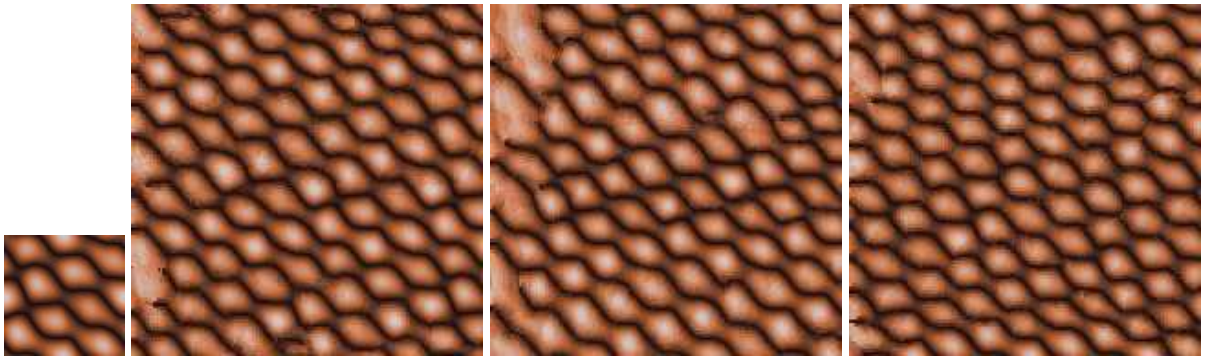


(a) (b) (c) (d)

(e) (f) (g)

Obrázek 5.6 - Klasický přístup bez urychlení, vstupní textura (a) o rozměrech 64x64 je detailně strukturovaná. Na následujících obrázcích jsou výsledky syntézy textur s maskou 2(b), 3(c), 4(d), 5(e), 6(f), 9(g). Zdroj (a) je [5].

Do této doby byla známa pouze 1 maska shodná pro obě úrovně (jak aktuální úroveň L, tak úroveň předchozí L+1). Masky jsou tedy v podstatě dvě a každou z nich je možné si nastavit (viz obr.5.7).

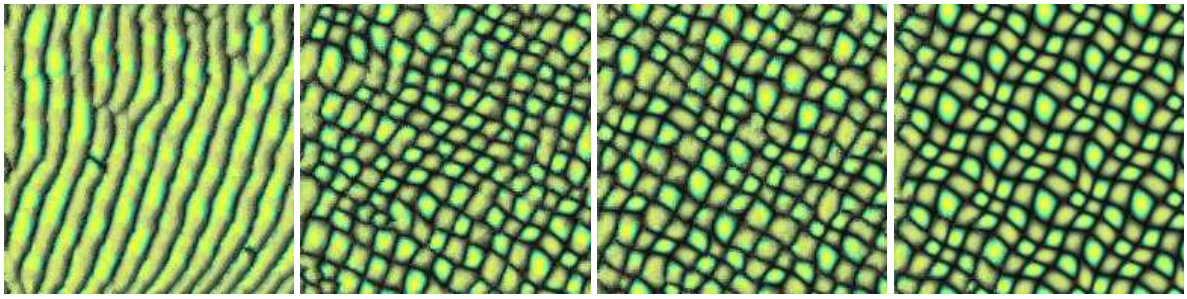


(a) (b) (c) (d)

Obrázek 5.7 - Byla použita klasická metoda bez urychlení s maskami 6;6(b), 6;4(c) a 6;8(d). Zdroj (a) je [5].

5.2.3 Kriterium kvality - výška pyramidy

Výška pyramidy také velmi ovlivňuje výsledek. Pokud je použitá úplná pyramida, obrázek dosáhne nejlepšího zobrazení (viz obr.5.8). Je to proto, že úrovně pyramidy představují detaily malířova obrazu. Nejdříve se syntetizuje nejvyšší úroveň, malíř kreslí pouze obrysy, a čím je úroveň nižší, tím malíř zachycuje menší detaily.

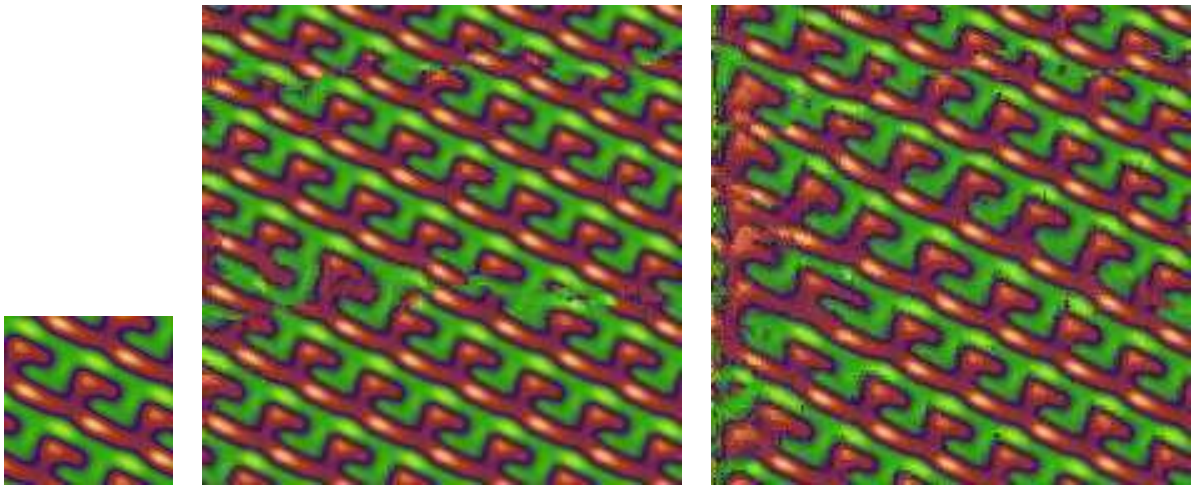


(a) (b) (c) (d)

Obrázek 5.8 - Syntetizování vzorové textury [] 64 x 64 za použití klasické metody bez urychlení s výškou pyramidy 1(a), 2(b), 3(c), 4(d) a maskou 3. Zdroj (a) je [5].

5.2.4 Kriterium kvality - použitý algoritmus

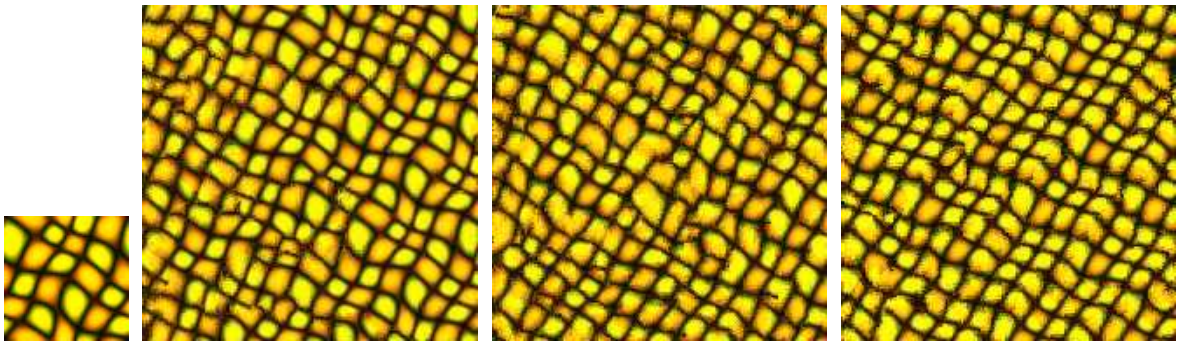
Způsob výpočtu (algoritmus) je posledním zde uvedeným faktorem ovlivňujícím kvalitu výsledného obrázku. Je zřejmé, že za daných vstupních parametrů bude mít nejlepší výsledek klasická metoda bez jakéhokoliv urychlení. Metoda „Omezení Intervalem“ s rozmezím kolem ±350 podává v řadě případů srovnatelné výsledky (viz obr.5.9).



(a) (b) (c)

Obrázek 5.9 - Porovnání výsledných obrázků klasické metody bez urychlení (b) a „Omezení intervalem“ s intervalem 350 (c). Zdroj (a) je [5].

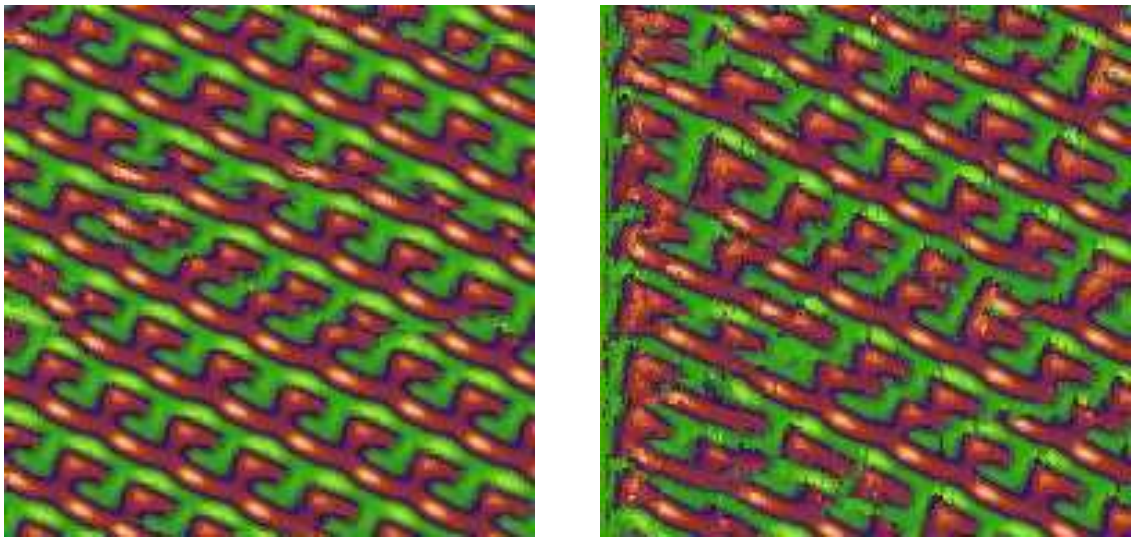
Na třetím místě, co se kvality týče, se umístila metoda FW-PLI, jejíž výsledky jsou velmi uspokojivé, ale struktura neodpovídá struktuře vzorku a není vždy pravidelná (viz obr.5.10), zvětšením masky se tomu dá částečně zabránit.



(a) (b) (c) (d)

Obrázek 5.10 - Srovnání výsledků syntézy pomocí metody klasické s maskou 3(b), FW-PLI s maskou 3(c) a FW-PLI s maskou 4(d). Zdroj (a) je [5].

S výsledky této metody jsou srovnatelné výsledky metody kombinované. Je to postup využívající, jak urychlení pomocí FW-PLI, tak pomocí „Omezení intervalem (při rozmezí kolem ±330) (viz obr.5.11).



(a) (b)

Obrázek 5.11 - Porovnání výsledných obrázků metody FW-PLI(a) a kombinované FW-PLI & IL(330)(b). Zdroj (a) je [6].

5.3 Porovnání výsledků s prácemi ostatních

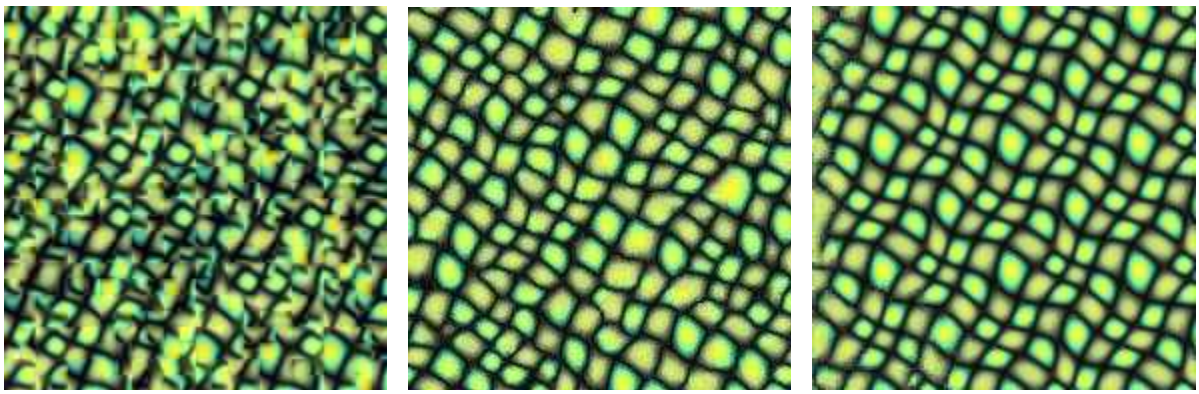
Algoritmus této bakalářské práce je postaven na metodě, práci a poznatcích Li-Yi Weie a Marca Levoye, proto by bylo dobré srovnat výsledky jejich a této metody syntézy textur viz figure. Aby srovnávání výsledků bylo objektivnější, budou se jej týkat i výsledky ostatních prací.

Základem porovnání metod budou dvě kritéria čas výpočtu a kvalita. Následující tabulka 5.1 obsahuje časy různých metod. Na následujících obrázcích je porovnána metoda této bakalářské práce

s metodou Li-Yi Weye a Marca Levoye (parametry odpovídají tabulce) a s metodou De Boneta (viz obr.5.12), algoritmy jsou bez jakékoliv urychlení.

Metody:	Parametry:	Čas výpočtu:
Metoda této BP - bez urychlení	64 -> 192, maska 3	187 seconds
Weie a Lewoy - bez urychlení	64 -> 192, maska 4	503 seconds
Efros a Leung	64 -> 192, ???	1941 seconds
Metoda této BP - IL(350)	64 -> 128, maska 3	43 seconds
Metoda této BP - FW-PLI	64 -> 128, maska 4	26 seconds
Metoda této BP - FW-PLI & IL(330)	64 -> 128, maska 4	14 seconds
Weie a Lewoy - TSVQ	64 -> 128, maska 4	24 seconds

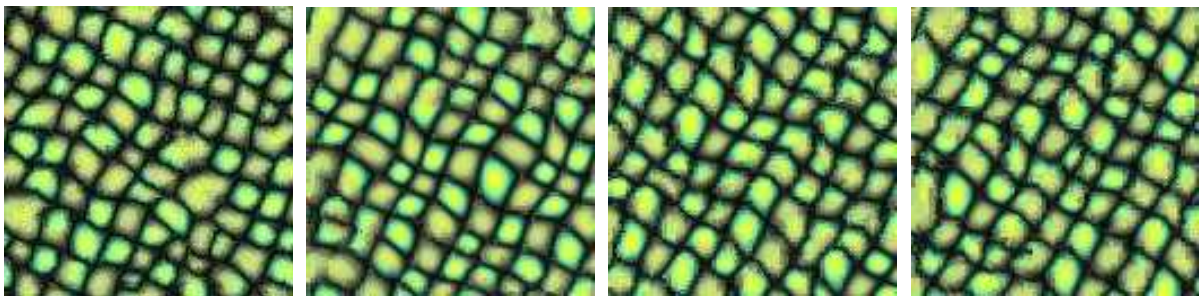
Tabulka 5.1 - V tabulce jsou uvedeny výsledky času průběhu jednotlivých metod a jejich parametry (64 -> 192 znamená vstupní textura s rozlišením 64x64 a výstupní textura 192x192).



(a) (b) (c)

Obrázek 5.12 - Porovnání výsledné textury metody De Boneta (a), Li-Yi Weye a Marca Lewoye (b) a metody této bakalářské práce(c). Zdroj (a,b) je [5].

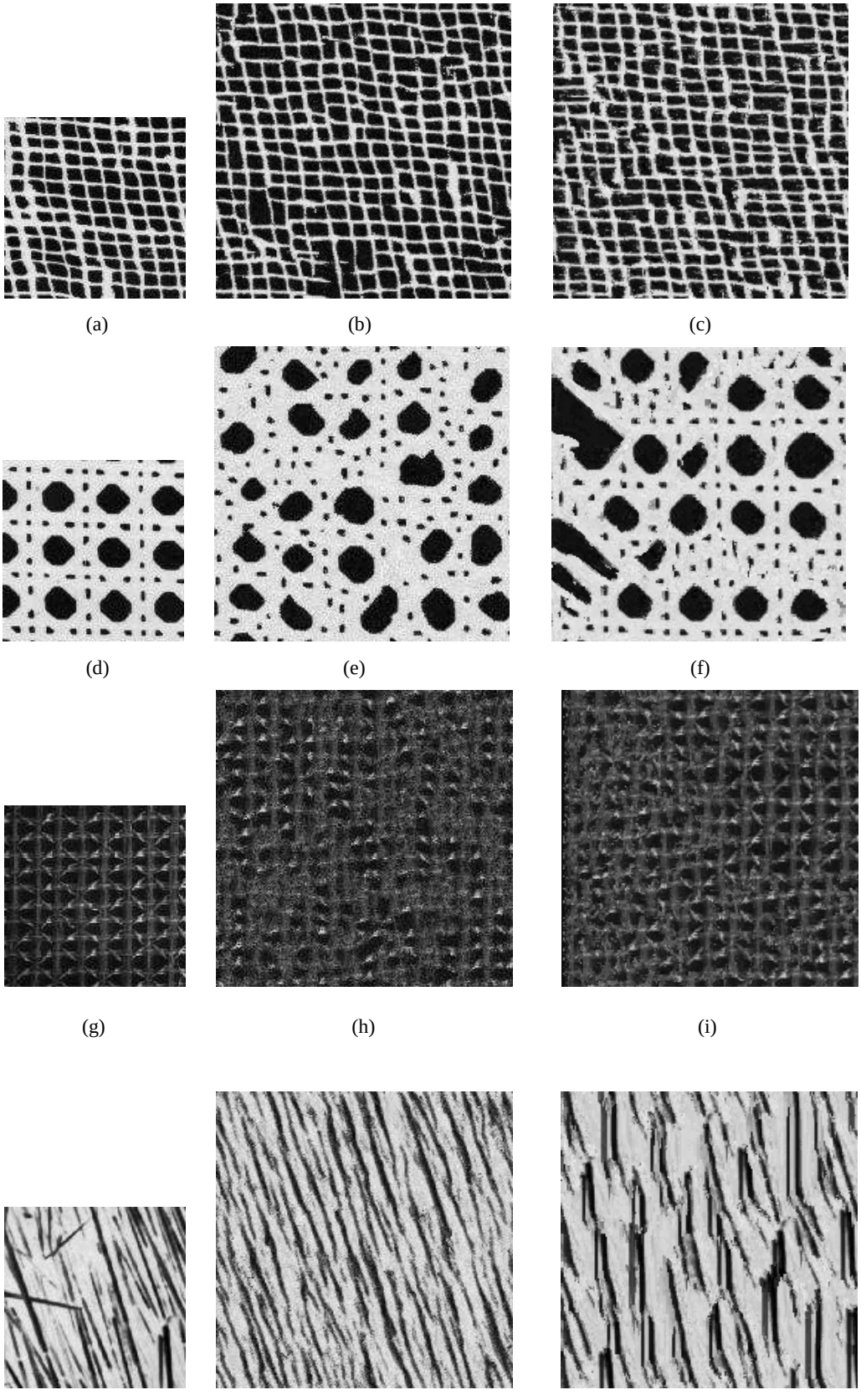
Tabulka 5.1 vypovídá o tom, že nejrychlejší z uvedených metod je kombinovaná metoda této bakalářské práce FW-PLI & IL(330) (kombinace dvou přístupů Find Wa by Previous Level Information a Interval Limitation). Protože doby výpočtu algoritmu této bakalářské práce a ostatních algoritmů byly naměřeny na rozdílně výkonných procesorech, může být TSVQ Li-Yi Weye a Marca Lewoye rychlejší než kombinovaná metoda FW-PLI & IL(330). Jelikož urychlení algoritmů způsobují ztráty na kvalitě, dle obrázku 5.13 metoda mojí práce syntetizuje nejlepší textury ze zde uvedených metod (viz obr.5.12).



(a) (b) (c) (d)

Obrázek 5.13 - Porovnání výsledných obrázků metody Li-Yi Weye a Marca Lewoye s použitím TSVQ (a) a metody této bakalářské práce za použití urychlení Interval Limitation s intervalem 350(b), FW-PLI (c) a kombinované metody FW-PLI & IL s intervalem 330(c). Zdroj (a) je [6].

Jako poslední bod testování, se budou porovnávat výsledné textury nejrychlejších dvou postupů TSVQ Li-Yi Weye a Marca Lewoye a kombinovaný FW-PLI & IL(330) (viz obr.5.14).



Obrázek 5.14 - Porovnání výsledků TSQV (b)(e)(h)(k) a FW-PLI & IL(330) (c)(f)(i)(l). Zdroj (a,b,d,e,g,h,j,k) je [5].

6 Závěr

Práce na této bakalářské práci mi přinesla mnoho zkušeností z oblasti počítačové grafiky, jsem s výsledky své práce nadmíru spokojen. Díky přesnějšímu vzorci pro výpočet bodů generované textury bylo dosaženo lepších výsledků než v metodě Li-Yi Weye a Marca Lewoye, na níž celá moje metoda stojí.

Obohatil jsem algoritmus o dvě metody Find Way by Previous Level Information a „Omezení Intervalem“. Provedl jsem testování a porovnávání metody na široké škále textur, pokud to byly textury s nějakou hierarchickou strukturou, tak má metoda dosáhla ve většině případů kvalitních výsledků a pokud se jednalo o textury nestrukturované (náhodné), výsledky syntézy byly někdy uspokojivé, jindy ne.

Program jsem vytvořil jako konzolovou aplikaci v C++, jako aplikaci .NET Windows Forms v C++ a také jako .NET Windows Forms v jazyce C#. Aplikaci v C# jsem pro nedostatečnou výkonnost zavrhl. Získal jsem další dovednosti v prostředí .NET Microsoft Windows, seznámil jsem se s knihovnou CImg pro práci s obrázky, které jsem využíval při programování v C++. V jazyce C# byla tato knihovna nahrazena referencí Systém::Drawing. Zjistil jsem, že pro programování výpočetně náročných aplikací je vhodnější použít méně pohodlný jazyk C++.

V budoucnu by se tato metoda syntézy textur mohla obohatit o další postupy, které by celý výpočetní proces ještě více urychlily. Díky své rychlosti a dobré efektivitě by se mohla dobře využít pro vývoj aplikace, která by retušovala staré fotografie nebo opravovala snímky nekvalitně zaznamenaných nebo starých videonahrávek. Dále by se mohla rozvinout také do oblastí 3D grafiky, kde by z malých vzorků vygenerovala textury pro jednotlivé 3D objekty.

Literatura

[1] Saphiro, Linda G., Stockman, G.,,: *Computer Vision.* Prentice-Hall, New Jersey, 2001. ISBN 0-13-030796-3

[2] Forsyth David,A., Ponce, J.,,: *Computer vision a modern approach.* Prentice-Hall, New Jersey, 2003. ISBN 0-13-191193-7

[3] Wei, Li-Yi, Lewoy, M.: Fast Texture Synthesis using Tree-structured Vector Quantization [online]. Stanford University, SIGGRAPH 2000, Dostupné na URL:< http://graphics.stanford.edu/papers/texture-synthesis-sig00/texture.pdf >

[4] Paget, R., Longstaff, D.,: Texture Synthesis via a Noncausal Nonparametric Multiscale Markov Random Field [online]. Department of Electrical and Computer Engineering, University Of Queensland 1998, Dostupné na URL:
< http://www.texturesynthesis.com/papers/Paget_IP_1998.pdf>

[5] Wei, Li-Yi, Lewoy, M.: Texture Synthesis examples[online]. Dostupné na URL:
< http://www-graphics.stanford.edu/projects/texture/demo>

[6] CGTextures.: Dostupné na URL:
< http://www.cgtextures.com>

[7] Qing Wu, Yizhou Yu: Feature Matching and Deformation for Texture Synthesis [online]. University Of Illinois at Urbana-Champaign, Los Angeles 2004, Dostupné na URL:
< http://www.cs.uiuc.edu/homes/yyz/publication/texsyn.pdf>

[8] Wikipedia, Dostupné na URL:
< http://www.wikipedia.org>

[9] Donadio, M: How To Generace Gaussian Noise, Dostupné na URL:
< http://www.dspguru.com/howto/tech/wgn.htm>

[10] Tschumperlé,D.: The CImg Library. Dostupné na URL:
< http://cimg.sourceforge.net/index.shtml>

[11] Kršek,P.: Základy počítačové grafiky – Studiknί opora., VUT Brno 2006. Dostupné na URL:<https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IZG-IT/texts/izg_opora.pdf>

Přílohy

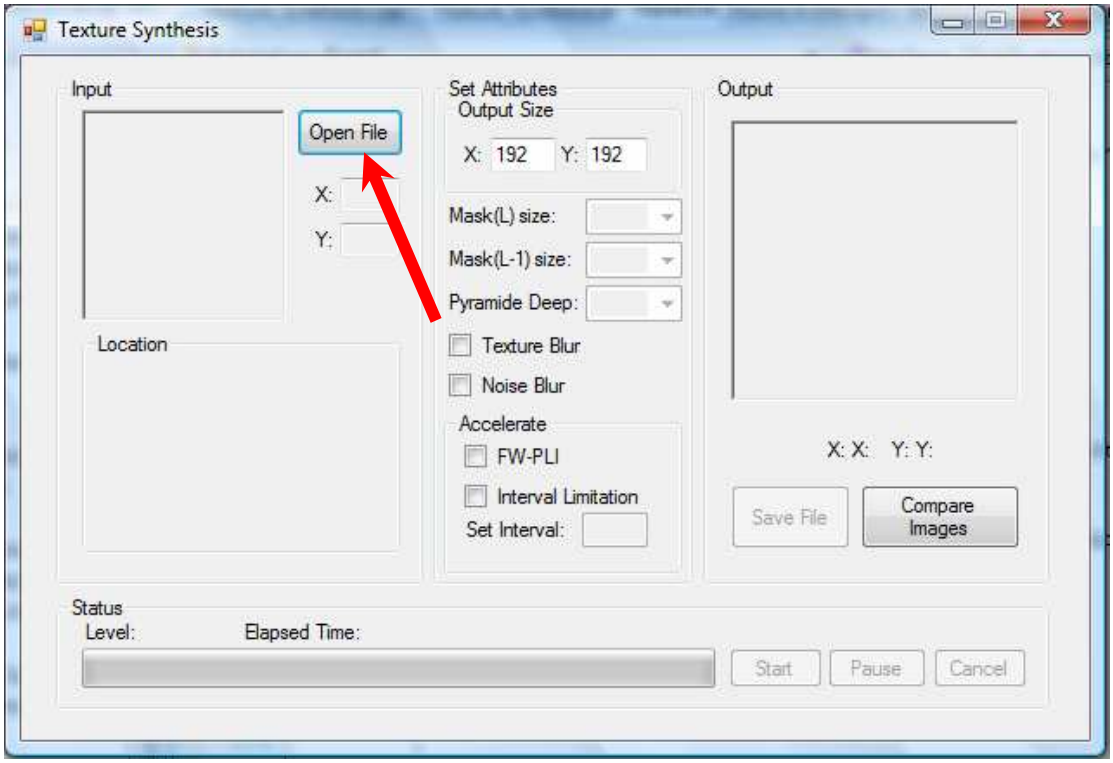
Příloha A. CD/DVD

Obsahuje zdrojový text v elektronické podobě písemnou zprávu,která je ve formátu PDF, zdrojový tvar písemné zprávy, úplnou dokumentaci, zdrojové texty programů a programy ve spustitelné formě.

Příloha B. Tutoriál k aplikaci .NET Windows Forms C++

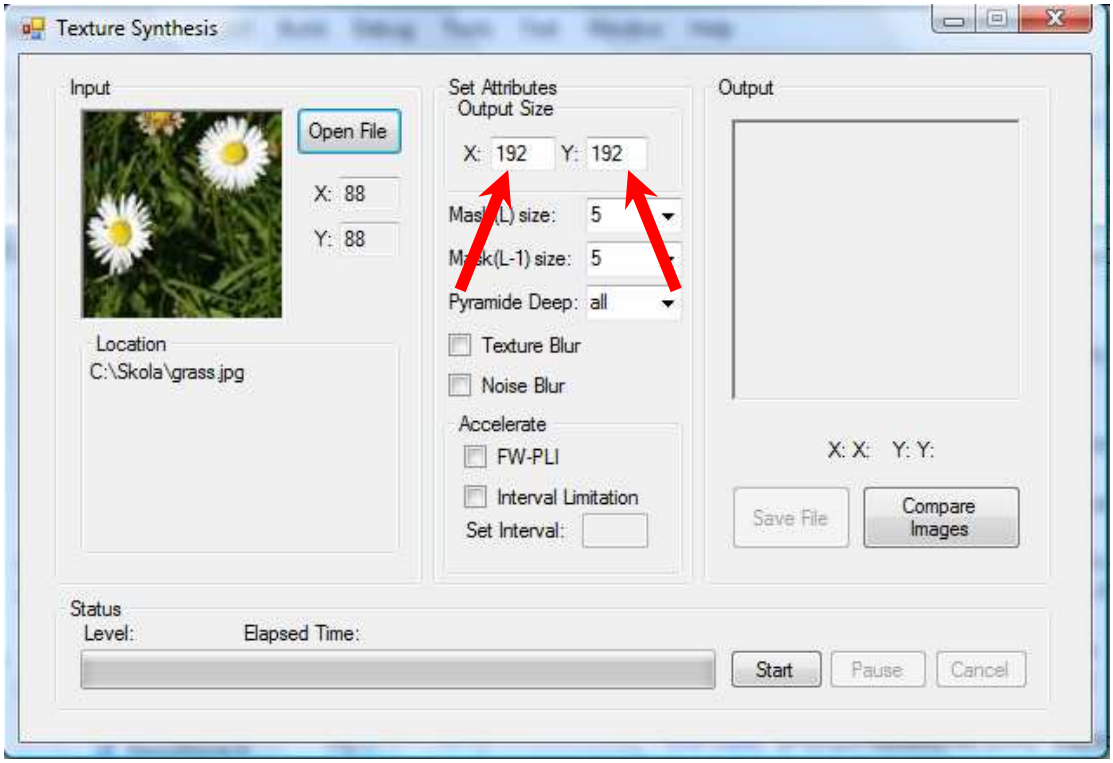
Po spuštění aplikace se objeví hlavní okno, kde se nastavují jednotlivé parametry syntézy:

1.) Kliknutím myši na tlačítko **Open** se vybere textura, kterou chceme syntetizovat (viz obr.B.2).



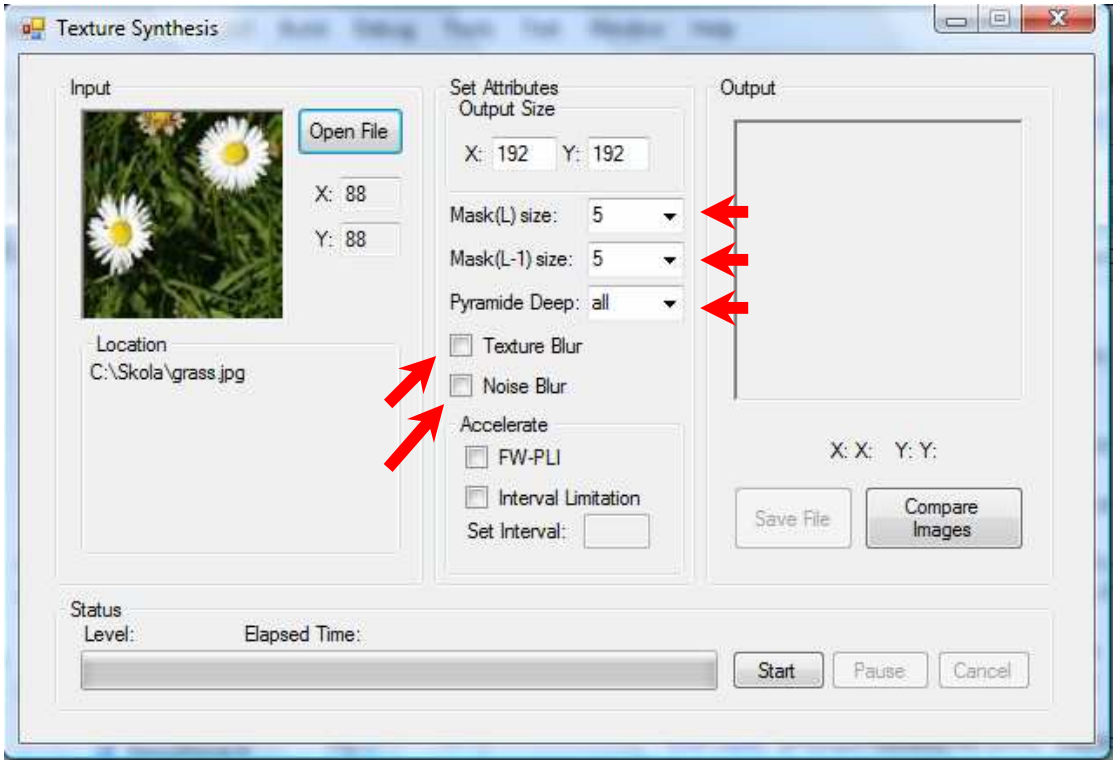
Obrázek B.2 – Tutoriál (1/10)

2.) Nastavíme parametry výstupního obrázku **Output Size** (**X** a **Y**) (viz obr.B.3).



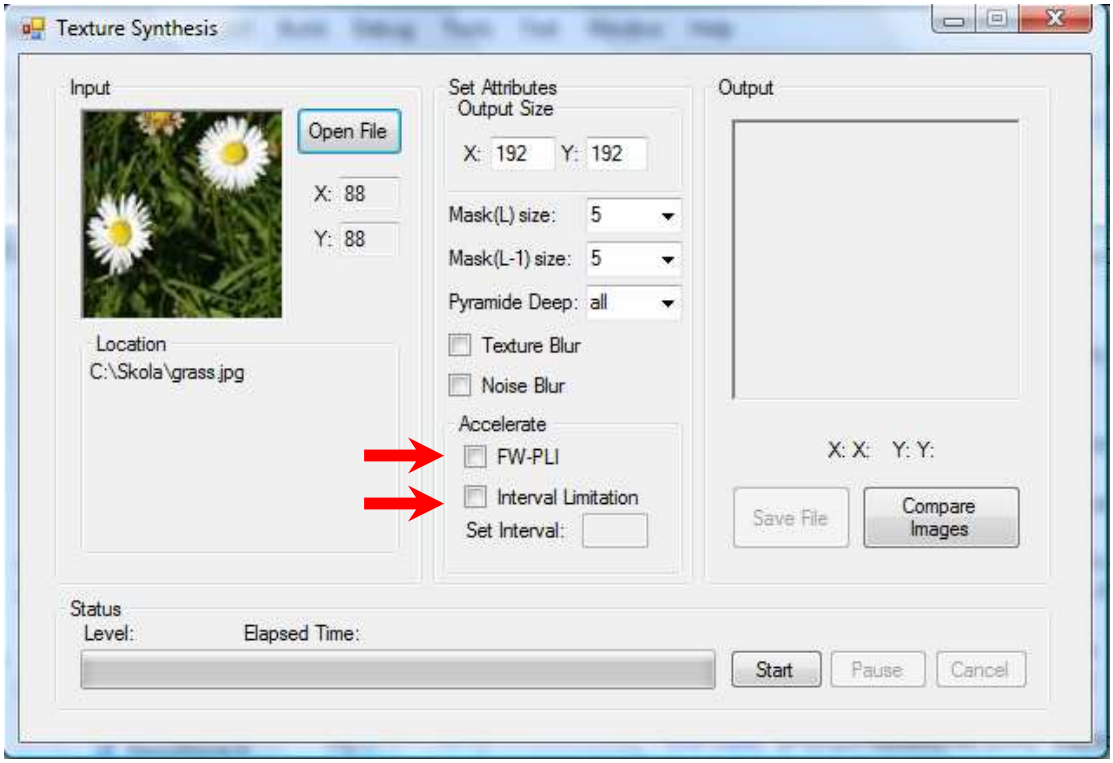
Obrázek B.3 – Tutoriál (2/10)

3.) Nastavíme parametry masky(L) a (L-1), zvolíme hloubku (výšku pyramidy), a nastavíme Gaussovy filtry při stavbě pyramidy vzorku textury (**Texture Blur**) a při stavbě pyramidy obrázku s náhodným šumem (**Noise Blur**) (viz obr.B.4).



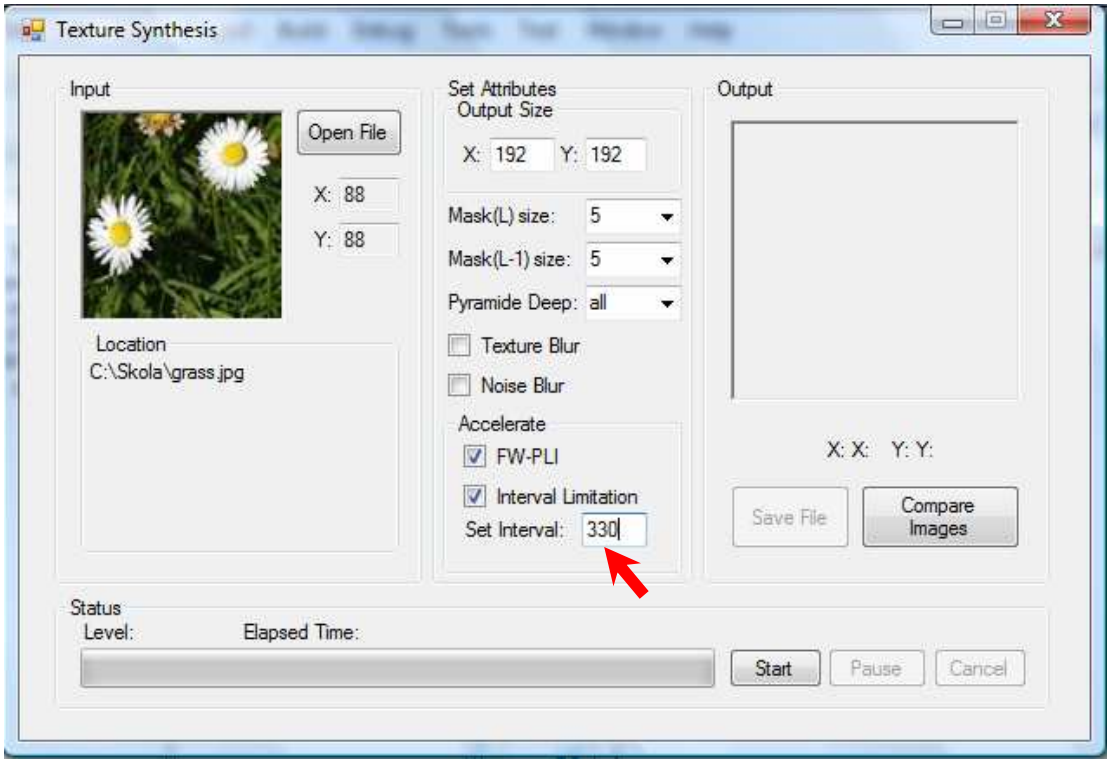
Obrázek B.4 – Tutoriál (3/10)

4.) Výpočet urychlíme aktivací FW-PLI nebo Interval Limitation („Omezení intervalem“) (viz obr.B.5).



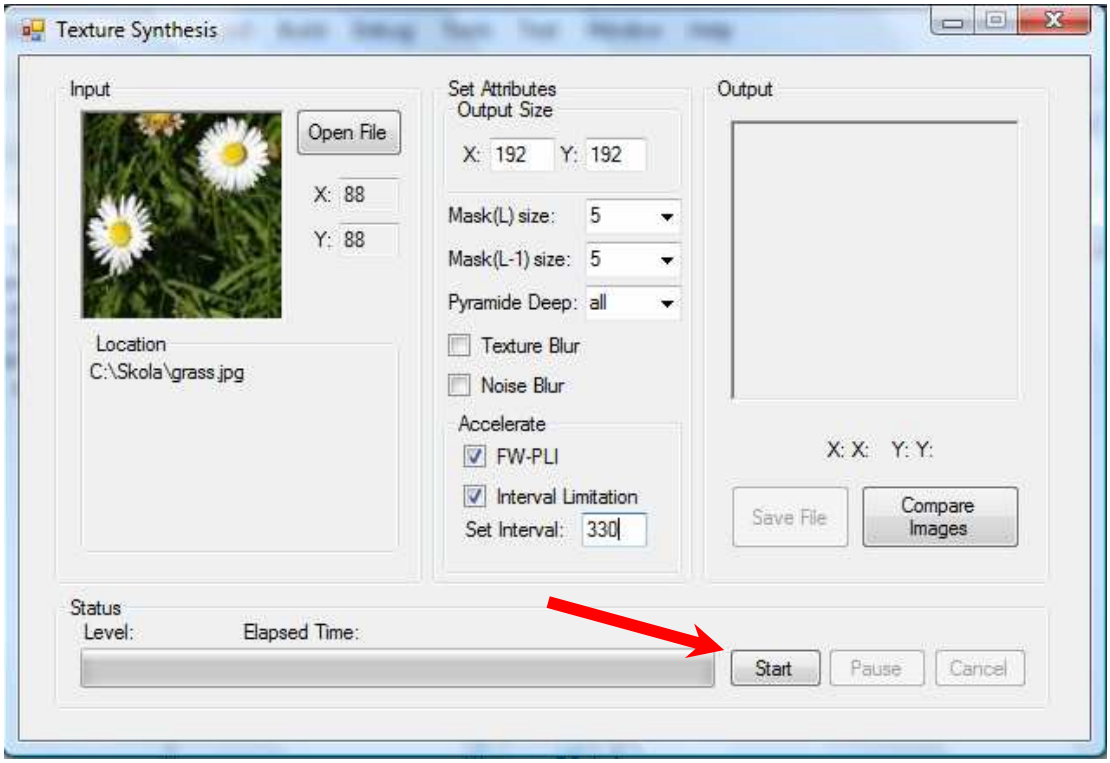
Obrázek B.5 – Tutoriál (4/10)

5.) Pokud volíme urychlovací algoritmus Interval Limitation, je nutné nastavit interval Set Interval (optimální hodnota 300 – 350) (viz obr.B.6).



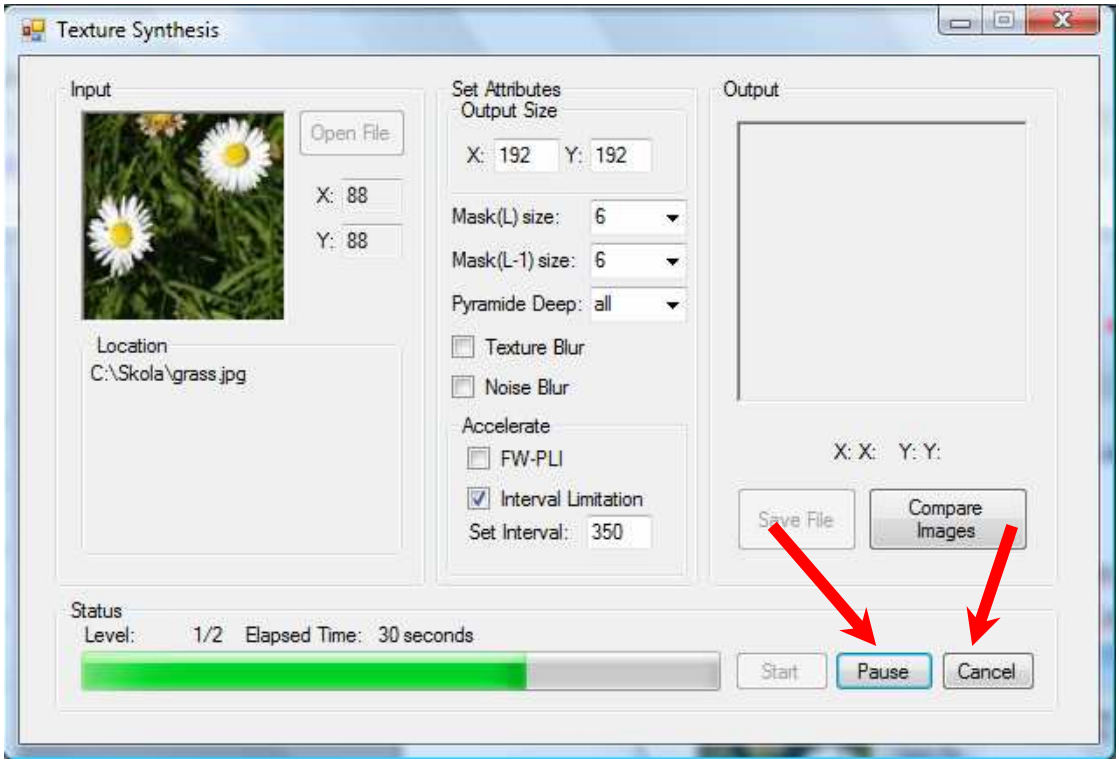
Obrázek B.6 – Tutoriál (5/10)

6.) Syntézu spustíme tlačítkem Start (viz obr.B.7).



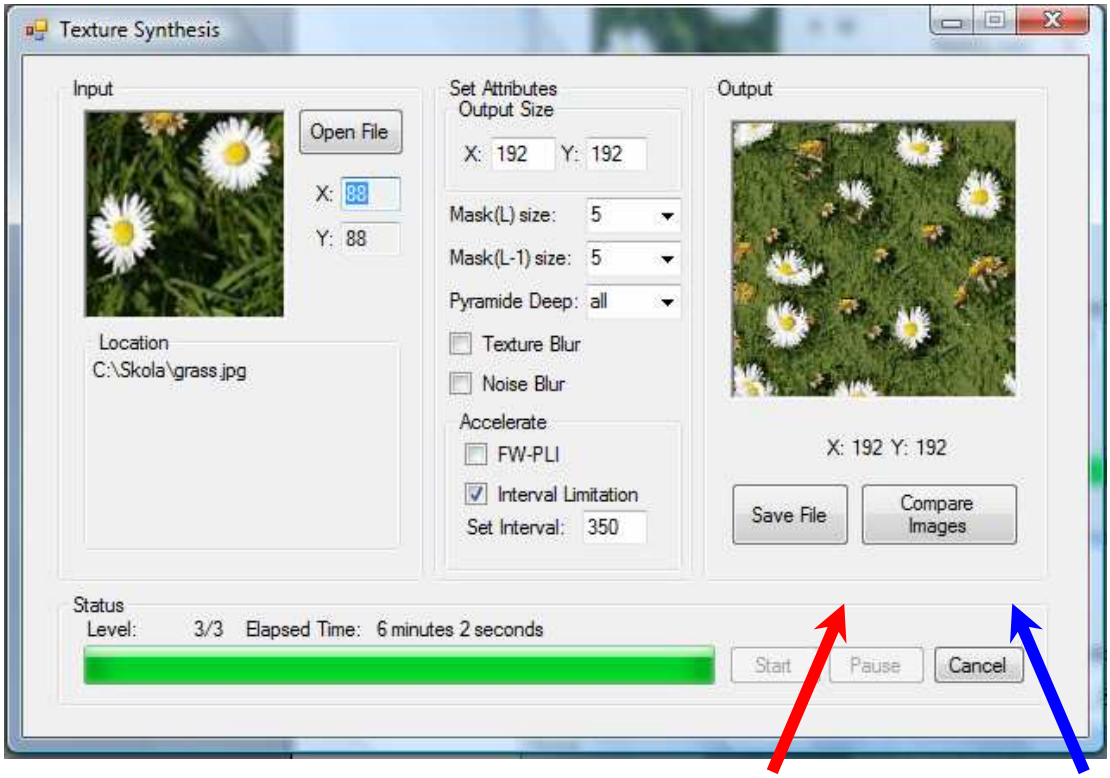
Obrázek 4.7 – Tutoriál (B/10)

7.) V průběhu můžeme proces syntézy pozastavit nebo zrušit pomocí **Pause** a **Cancel** (viz obr.B.8).



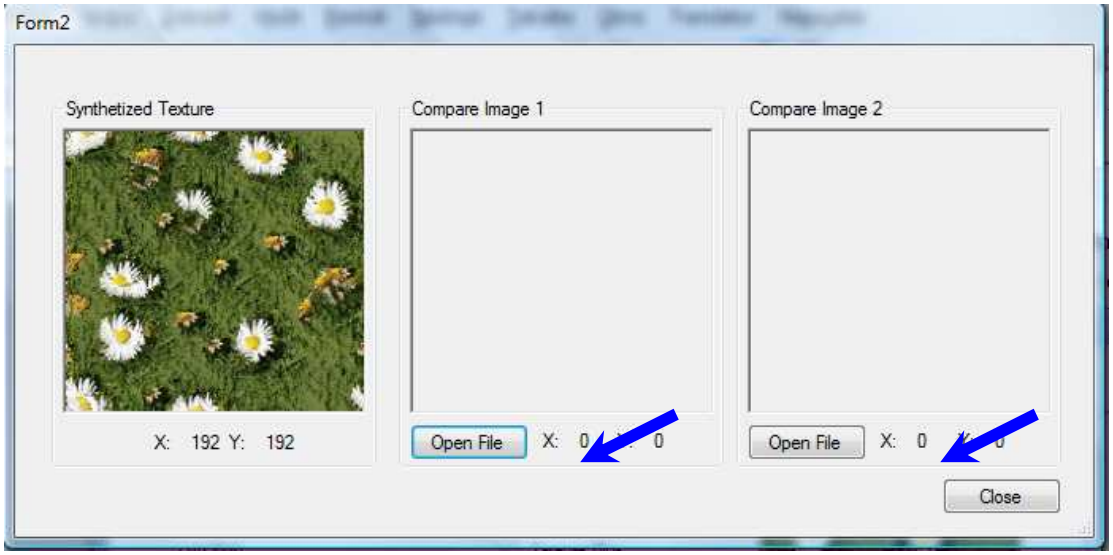
Obrázek B.8 – Tutoriál (7/10)

8.) Po dokončení můžeme výsledek uložit **Save File** a křížkem program ukončit, nebo výsledek porovnat s jinými obrázky kliknutím na **Compare Images** (viz obr.B.9).



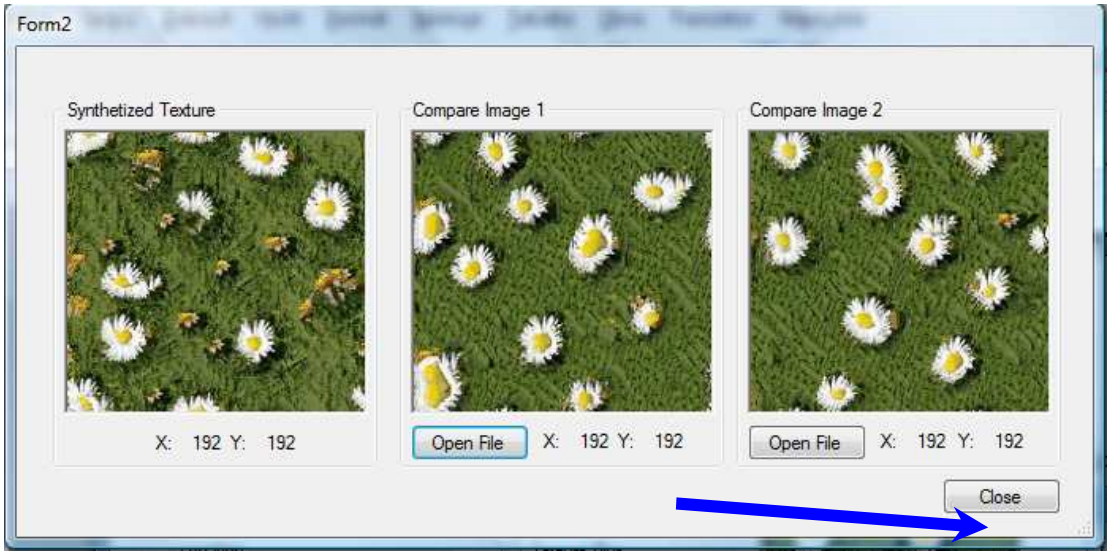
Obrázek B.9 – Tutoriál (8/10)

9.) Po kliknutí na Compare Images se objeví druhý formulář, kde můžeme porovnávat výsledek, který se objeví nalevo, s dvěma dalšími obrázky tak, že klikneme na tlačítko Open File.



Obrázek B.10 – Tutoriál (9/10)

10.) Druhý formulář uzavřeme **Close** a křížkem ukončíte aplikaci (viz obr.B.11).



Obrázek B.11 – Tutoriál (10/10)